Technical Report 1023

The Use of Grouping in Visual Object Recognition

David W. Jacobs

MIT Artificial Intelligence Laboratory

This blank page was inserted to preserve pagination.

The Use of Grouping in Visual Object Recognition

David W. Jacobs

Revised Version of a Thesis submitted to the Department of Electrical Engineering and Computer Science, January, 1988, in partial fulfillment of the requirements for the degree of Master of Science.

iii

Abstract

This report explores the use of grouping in object recognition by computational systems. Many recognition systems in the past have performed an undirected search through the space of different segmentations of an image in order to recognize objects. This approach leads to significant problems of computational complexity and accuracy. The process of grouping determines the sections of an image most likely to come from a single object. This can tell a recognition system which segmentations of the image to consider first, improving both its speed and accuracy.

The report describes a particular recognition system called GROPER. GROPER performs grouping by using distance and relative orientation constraints that estimate the likelihood of different edges in an image coming from the same object. The thesis presents both a theoretical analysis of the grouping problem and a practical implementation of a grouping system. And it discusses the relevance of this theory of grouping to human psychology. GROPER also uses an indexing module to allow it to make use of knowledge of different objects, any of which might appear in an image. We test GROPER by comparing it to a similar recognition system that does not use grouping.

Thesis Supervisor: Prof. W. Eric L. Grimson

Acknowledgments

General Motors generously provided me with support during much of the time I worked on GROPER. This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Office of Naval Research University Research Initiative Program under Office of Naval Research contract N00014-86-K-0685 and in part by the Advanced Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K-0124.

I'd like to thank my advisor, Eric Grimson, for all the time he spent discussing this research with me. He provided many useful ideas, and much good advice about how to proceed. Whitman Richards gave his time freely to help me. His stress on the formal analysis of grouping constraints proved an extremely valuable influence.

Many people have contributed greatly to my understanding of this work, including Todd Cass, Dave Clemens, Liz Edlind, David Lowe, and Jim Mahoney.

I'd like to thank all the members of the AI lab who provided the computational environment that made this work possible. I used many tools without even knowing who was responsible for them. I'd especially like to thank Eric Grimson and Tomás Lozano-Pérez for allowing me to use some of the code from their recognition system, and Keith Nishihara and Noble Larson for developing the hardware/software environment I used for image processing.

I'd like to thank my Mother, Father, and Sister, for all of their support.

And thanks to Liz, who not only provided tremendous support while I worked on this, but who also put up with me.

Contents

XI

Chapter 1

Introduction

1.1 Why Grouping?

This thesis argues for the use of grouping in model-based, visual object recognition. According to this thesis, a computer attempting to recognize objects in an image should begin by determining what sections of the image seem likely to have come from a single object. We will call this process grouping. Only after performing grouping should the computer system try to determine which objects might have produced these sections of the image. This thesis follows the work of Lowe[18], who built the first computational recognition system to explicitly make use of grouping.

By model-based, visual object recognition I mean the process we employ in day to day life, such as when I look at my desk and recognize the stapler, the text book, and the piles of papers, failing to locate my pen. Grouping means deciding that some segment of this cluttered scene all comes from a single object. For example, suppose I notice that something silver, with a red blob at the end, is sticking out from under my notebook. Perhaps, before I recognize what that thing is, I decide that it is probably all one thing; that the silver part of the image and the adjacent red part of the image probably both came from just one object. That decision would belong to the grouping process, as I will use the term "grouping". I might then use the shape of the silver and red blobs, and their relationship, to deduce that the thing is a pencil. If so, I have recognized the pencil with the assistance of grouping. The reader will not necessarily find it obvious that we actually use grouping when we recognize objects; this example only illustrates how one might use grouping as part of the recognition process.

A more detailed example will help explain this process better. It will show how this thesis formulates the object recognition problem, and how grouping can play a part in a model-based recognition system. It will also demonstrate three advantages of using grouping

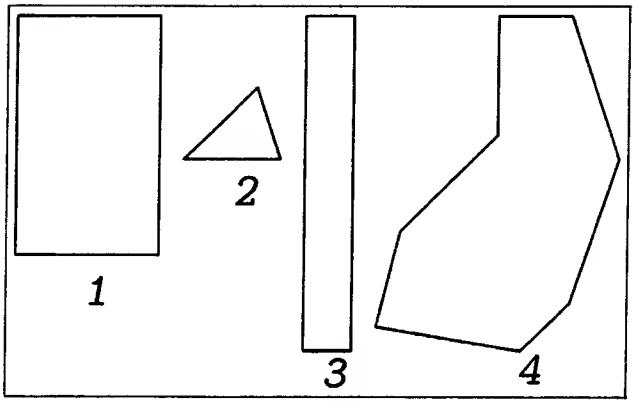

Figure 1.1: Models of Geometric Shapes

to recognize objects. First of all, grouping can help us to avoid mistakes we might otherwise make. Secondly, it can speed up recognition by reducing the number of possibilities we must consider. And thirdly, it can provide more information to use in determining what objects we must consider. A comparison between recognition that uses grouping and recognition that does not make use of grouping will demonstrate these advantages.

Suppose we wish to use a computer to recognize geometric shapes. The computer will begin with two kinds of information. First of all, it will begin by knowing about some geometric shapes, which it will try to recognize. It might, for example, start off knowing about the shapes in figure 1.1. Secondly, the computer will receive an image that contains some of these shapes, such as the image shown in figure 1.2. This example, though simple, will demonstrate some characteristics of more complicated object recognition problems. So in this image, some shapes overlap others, covering up part of their perimeter. And sections of the perimeter of some shapes do not appear, just as parts of an object may blend into the background, producing an indistinct outline. The problem then has these two inputs: models of objects to recognize, and an image that may contain some of these objects, partially obscured, in unknown positions.

For its output, we want the computer to locate the shapes that it knows about as quickly and accurately as possible.

We can do this by interleaving a grouping phase, and a recognition phase. We first use grouping to determine what edges in the image seem to have come from a single geometric shape. Then in the recognition phase, we determine which, if any, of the known shapes could have produced the edges that the grouping phase has lumped together. Based on the results, we will decide how to proceed.

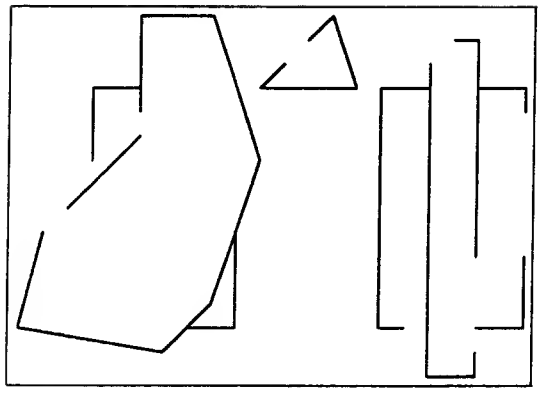

Figure 1.2: A Sample Scene

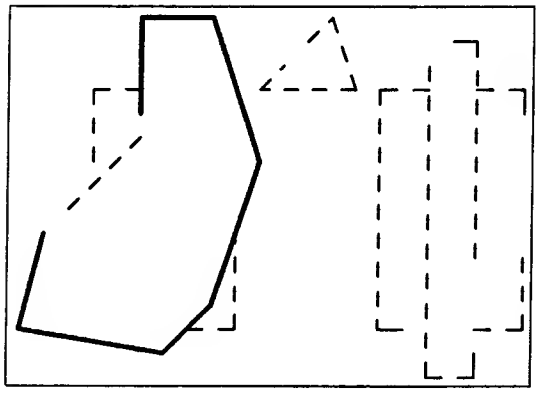

Figure 1.3: The highlighted edges look like they all came from the same object.

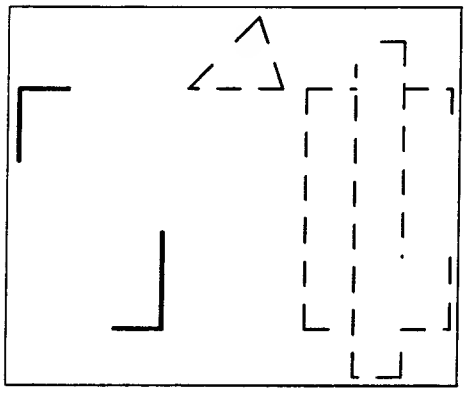

Figure 1.4: These edges also appear to have come from a single object.

So we might begin by deciding that the edges highlighted in figure 1.3 seem to have all come from a single object. (How we make decisions like that will take up a large portion of this thesis). We then compare these edges to the shapes in figure 1.1, and successfully identify shape number 4. We might then decide that, of the remaining edges in the image, the ones highlighted in figure 1.4 appear to have all come from a single object. We compare these edges to the shapes we know about, and conclude that they must have come from shape number 1. This vague description of an approach to model-based object recognition highlights two main problems on which this thesis will concentrate: how we decide which groups of edges all came from a single object, and how we decide from which object a group of edges came.

If we can solve these two problems, we will have a system with some considerable advantages over other approaches to model-based object recognition. To show this, I will propose a straw man recognition system that makes no use of grouping at all. Most computational, model-based object recognition systems make use of some type of grouping, at least implicitly. But without any grouping system, we would have to search randomly through the space of all possible collections of edges, until we found one in which all the edges really did come from a single shape. We might start out selecting two edges at random from the image. We would then compare these edges to our shape models. If we could not find any known shape that could have produced these two edges, we would give up on them and try another pair of edges. On the other hand, suppose we knew of three different shapes that could have produced that randomly chosen pair of edges. Then we would randomly select a third edge, and see if any of the three objects might have produced that edge as well. We would continue in that way, until we came up with some collection of edges that only one shape might have produced, and in that way succeed in recognizing a shape in the image.

The next chapter will discuss these two approaches to model-based recognition in more detail. But the brief example given allows us to discuss three significant advantages of having a grouping system.

First of all, a good grouping system can quickly lead us to the right answer. Instead of searching at random for good combinations of edges, a grouping system is designed to tell us which combinations are most likely all to have come from a single object. We can perform grouping in less time than it would take to compare all different combinations of edges to our models, because grouping does not depend on the objects that we are looking for, and because it is a local visual process.

Secondly, grouping can save us from making certain kinds of mistakes. Suppose, without the assistance of grouping, we ended up wondering which object might have produced the edges highlighted in figure 1.5. When we compared these edges to our models of shapes, we would find that only shape 1 might have produced them all. This would provide us

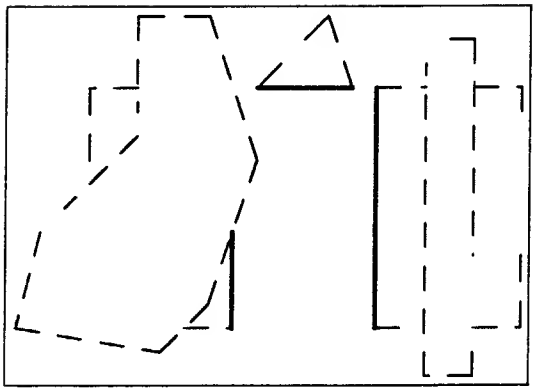

Figure 1.5: If we try collections of edges at random in our search for objects we might consider the edges highlighted. Of the four objects we know about, only object one could have produced all these edges.

with a strong reason to locate shape 1 at that position in the image. We would be wrong. Intuitively, people do not make that mistake because those particular edges do not look as if they all came from the same object. And there are additional edges in the image that seem to go with each of the three edges in this combination.

This example may seem a bit contrived. After all, what are the chances that some randomly chosen edges in an image, which do not really all come from the same object, will nonetheless look just like some object we know about? In fact, if we know about thousands of different objects, all of which have flexible parts, there is a good chance that for any small random collection of edges we can find some object we know about, in some position that might have produced those edges. Without grouping, this will complicate the recognition process and increase the chances of errors occurring.

Thirdly, grouping provides a more advanced starting point for the recognition process. When we recognized shape 1 from the edges highlighted in figure 1.4, we began with four edges provided for us by the grouping process, and only then looked at our models to decide which shape might have produced those edges. Because we already had a collection of four edges, we could use the relationship between these edges to index into our list of known shapes and weed out the ones that could not possibly match. For example, we might notice that those four edges produce two ninety degree corners. Of the four shapes we know of, only two contain two ninety degree corners. In this way we can quickly find that we need to consider only half of the shapes.

Grouping has the potential to improve the speed and accuracy of a model-based object recognition system. It can speed things up by leading us quickly to large collections of edges

Figure 1.6: A picture of six objects.

that all come from a single object. This will cut down on the number of false possibilities we must try out, and at the same time allow us to index into a library of different objects that might have produced the edges, cutting down on the number of objects we must consider. Grouping can improve the accuracy of a recognition system by providing additional evidence for the existence of a suspected object in the image. Instead of only considering whether an object could have produced some collection of edges, we may also consider whether in fact those edges really seem to have all come from the same object. As we address complicated recognition tasks, we will need both of these benefits more and more.

Figures 1.6 and 1.7 show the results of a grouping-based recognition system. It successfully finds all the objects in the image, considering a total of only 8 different collections of edges in the process.

1.2 The Structure of the Thesis

The rest of this thesis will elaborate the points hinted at in this introduction. Chapter 2 will provide a more complete overview of this work. Chapter 3 will present a theory of computation for grouping. To perform grouping, we would like to know the probability that some edges all come from a single object, given their relative positions. Due to the difficulty of accurately modeling the real world, this chapter presents a model of a simplified world. It then draws some conclusions about that simplified world which suggest ways of approximating the real world probabilities about which we would like to know.

Using this theory of grouping, chapter 4 will investigate the relationship between the suggestions for computational grouping presented in this thesis and human grouping per-

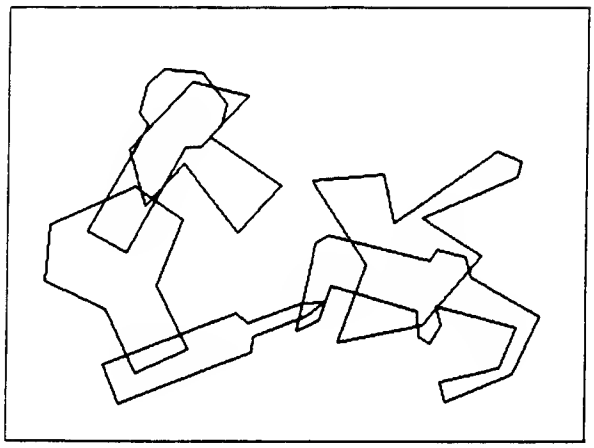

Figure 1.7: The objects a grouping based recognition system finds in the previous image. To find these six objects, it had to consider only eight different collections of edges.

formance. When presented with images consisting of edges, people tend to see some of these edges grouped together. Chapter 4 will show that the theory of grouping correctly predicts some human grouping experiences.

Chapter 5 will describe a grouping algorithm based on this theory. This algorithm largely follows the lines suggested by the theory of grouping, but it does introduce some simplifications and fills in some of the holes left open by the theory.

Chapter 6 will discuss a model-based recognition system designed to work with grouping. It will describe a recognition module that indexes into a data base of different object models. That is, given a group of edges, this module looks in a table to see which objects might have produced those edges, instead of searching through the set of all possible matches of image edges to model edges.

Chapter 7 will briefly describe a verification step that can provide additional evidence for a match between some image edges and model edges. This verification step is simple, and based on similar work done by other researchers.

Chapter 8 will describe some work done by other researchers in model-based object recognition. In particular, it will point out the debt this work owes to that of David Lowe. In some respects, this work is an extension to Lowe's work on grouping. It will also point out that many other previous recognition systems work well because they implicitly use a certain amount of grouping. And it will discuss some other approaches to indexing.

Finally, chapter 9 will evaluate the results of the entire recognition system. It will show the performance of this system on real images. It will discuss how successfully the grouping system judges which edges should go together, and to what extent this grouping reduces the amount of computation needed to recognize objects.

Chapter 2

Overview

This chapter will discuss more precisely some of the notions mentioned in the introduction. It will describe a limited subset of the object recognition problem within which this thesis will operate. It will also more precisely define grouping and indexing, concepts that the previous chapter introduced at only an intuitive level. These definitions will allow a more careful argument in favor of grouping to develop, an argument that will contrast this approach with the approach taken in other object recognition systems. Finally, the end of the chapter will discuss what we may reasonably expect from a grouping system, explaining why we might hope to be able to build a recognition system using grouping.

2.1 What is Object Recognition?

Object recognition makes use of images of the world and knowledge about objects in order to locate objects. The knowledge we use in recognizing objects may range widely, from the specific shape of Jim Rice's face, to the fact that he often appears in the outfield at Fenway Park, but rarely at Shea Stadium. Images also contain a wide range of effects. Intrinsic qualities of objects produce differing light intensities throughout an image, but changes in illumination also effect these intensities. Often the perimeters of objects create sharp changes in intensity in an image. But such changes also occur due to shadows, specularities, and the texture or surface markings of objects. And sometimes objects' perimeters do not create sharp changes in intensity, because an object's surface material appears to blend into the background, or because another object stands between the first object and the viewer. We can characterize visual object recognition as the process that combines any knowledge of the world that the viewer may have with a continuous sequence of images of overlapping three dimensional objects subject to varying lighting conditions, in order to locate known objects in the world.

The wide variety of possible inputs to the object recognition problem make it difficult to solve. For that reason, this thesis will discuss an object recognition system, GROPER (GRouping-based Object PERception), that focuses on a more limited version of the recognition problem. Making progress on even a subset of the recognition problem may prove valuable because practical applications exist for solutions to limited recognition problems, and because progress on these problems may provide insight into how to address the more complete recognition problem. This thesis addresses recognition making use of only simple, well defined knowledge of the world, and handling images with only some of the complexity of the images that humans routinely handle.

Instead of making use of many kinds of knowledge of the world, GROPER will use only precise geometric knowledge about objects. It will know the exact size and shape of the objects it seeks, but it will not use any other kind of knowledge directly, including even the color or texture of the material that composes these objects. GROPER might, for example, know exactly what a specific stapler looks like. But it would not make use of some more general model to recognize a slightly different stapler. Moreover, one of GROPER's models of an object will only depict that object in a specific pose. So GROPER's model of a stapler might allow it to recognize the stapler in an ordinary position, but not when the top of the stapler is open so that it may be loaded with new staples. And GROPER would know how the stapler looked from a certain distance away, but would not recognize the stapler from a significantly greater or lesser distance.

This simplification of the problem will not effect the grouping portion of GROPER because grouping will not make any use of knowledge about particular objects. If an unknown object with flexible parts appears in the image, the grouping module will be just as likely to find edges that all come from that object as it will for a known object. But this limitation on knowledge of objects will greatly reduce the problems involved in building the recognition component of GROPER.

GROPER will also make use of two important simplifications in the types of images on which it will work. The research described in this thesis assumes that only two dimensional objects will appear in the image, such as flat machine parts. And the theory of grouping developed only addresses the problem of how to group together the occluding edges of objects that appear in the image, although GROPER does not assume that it knows on which side of an occluding edge the object lies, and which side is the background. Limiting the objects expected to two dimensions simplifies both the recognition and the grouping portion of GROPER. And the use of only occluding edges makes it much easier to decide what edges come from a single object.

Finally, GROPER will use only a single image of a scene when it looks for objects. We could gain much additional information about a scene from a series of images taken over a

period of time during which either the objects or the viewer moves.

GROPER does address some aspects of the recognition problem that many computational object recognition systems do not. GROPER will make use of knowledge of a library of different objects, many of which look quite similar, or have common subparts. And GROPER will recognize objects in scenes that contain about a dozen similar objects. These types of situations cause difficulties of accuracy and computational complexity. These difficulties will only grow more acute as researchers deal with recognition problems closer to the task that humans perform. They provide the motivation for GROPER's approach to recognition.

2.2 A Recognition Algorithm

GROPER's recognition algorithm makes use of three main components: grouping, indexing, and verification. Subsequent sections will describe these components in more detail, but first this section will explain how they fit together. In brief, grouping provides GROPER with groups of intensity edges that seem likely all to come from a single object. Indexing takes a group of intensity edges, and decides which edges from known objects might have produced these image edges. Verification then takes one of these matches between an object model's edges and image edges, and makes a final decision about whether that object might have produced those image edges.

GROPER uses these components in the following algorithm:

1. Extract straight line approximations to all the intensity edges in an image.

2. Using grouping, choose the group of edges thought most likely to have come from a single object.

3. Using indexing, determine which collections of model edges might match this group of image edges.

4a. If no modeled edges could match the image edges chosen, return to step two, using grouping to choose the next most promising group of edges.

4b. If only a few sets of model edges match these image edges, perform a verification step on these hypothesized matches. Extract the edges produced by any successfully recognized objects from the image, and return to step two.

4c. If a group could match many different sets of object edges, try to extend the group and narrow down the possibilities by using grouping to add more edges that seem to

come from the same object as the originally chosen edges. This will lead either to the situation in step four a or step four b.

This algorithm will help to explain why GROPER requires certain kinds of results from grouping, indexing and verification.

2.3 What is Grouping?

The introduction argued that grouping can help overcome problems of inaccuracy and complexity. This section will explain more clearly what grouping means. It will make three main points. First of all, this section will explain operationally what grouping does: what it begins with, and what it produces. Next, this section will explain how GROPER can perform grouping in a way that is independent of any particular library of objects for which it looks. Finally, this section will provide an overview of how GROPER does grouping.

First of all, we must distinguish the type of grouping that GROPER performs from a kind of feature detection that other authors sometimes refer to as grouping. Some authors use the word grouping to describe the process of making straight line approximations to the intensity edges in an image, or finding corners or other types of simple features in the intensity edges an image produces. This thesis does not mean that. In fact, GROPER uses straight line approximations to image edges as the input to its grouping component. By "grouping", we mean a process that attempts to locate collections of these straight lines that all seem to come from a single object.

The research described in this thesis only addresses the problem of grouping together the intensity edges produced by the perimeter of an object. We will not discuss, for example, how one might decide whether two different homogeneous chunks of an image probably come from the same object. For this reason, GROPER uses as input the results of an intensity edge detector, such as the Marr-Hildreth edge detector (Marr[20]). These edges make good candidates for the location of the perimeter of objects in an image.

Because this thesis only explores a serial algorithm for recognition, GROPER tries the best available groups of edges, one at a time, to see if each will help it to recognize an object. Several different factors make a group of edges good for a recognition system, but we will focus on only one of these, the fact that the greater the chances are that a group of edges comes from a single object, the more likely it is to lead GROPER to correctly identify an object. We might also select a group of edges because if the edges do, in fact, all come from a single object, then that group is also likely to lead quickly to the identification of that object. For example, we might feel that a particular group of two edges has a slightly greater chance of coming from a single object than a different group of five edges,

but that the group of five edges will provide us with more information which we can use to decide what object produced those edges. Or, we might want a grouping system to favor groups of edges especially suitable for our indexing scheme. Lowe[18] makes this a primary consideration in his grouping system. So three possible factors could influence one's criteria for grouping together edges: the edges' likelihood of coming from the same object, the extent to which they will narrow down our search if correct, and their usefulness in indexing. Of these three, GROPER emphasizes the first criteria; its output consists of the collection of edges deemed most likely to come from a single object.

Also, grouping must be relatively fast. Obviously, one way to perform grouping would be by looking for objects in the image, and grouping together all the edges that appear to come from the same object. To be a useful first step in object recognition, it must take much less time to group together edges than it would take to find objects without grouping.

Notice that to perform this function, grouping does not need to partition the edges in an image. The best collection of edges to try when looking for objects might combine an edge with some others, while another good possibility combines that edge with a different set of edges. So grouping does not need to divide image edges into non-intersecting sub-groups, but rather to produce an ordered collection of groups, which may partially overlap.

To determine the likelihood of a group of edges in an image coming from the same object, we must have a model of the world which produced those edges. In different kinds of worlds, we might approach grouping in different ways. In a world containing randomly oriented, flat squares, for example, we would only group together parallel or perpendicular edges. In a world with randomly shaped objects, on the other hand, we might not use the angle between edges at all in deciding their likelihood of coming from the same object. So the model we have of the world will have an important influence on the strategy we adopt for grouping.

The real world does not have a simple description. It has a huge variety of objects which appear, not according to some simple random distribution, but in locations influenced by a variety of factors. For example, objects need support or they fall down, and screwdrivers appear near hammers more often then they appear near elephants. We could not possibly come up with a complete model of the real world which accurately reflected all such facts.

So instead, GROPER makes use of a model of a simplified world of two dimensional, polygonal objects with convex subparts. The conclusions made about this world, however, seem to still apply well to the real world, as we will see later. The point we make here is that GROPER's model of the world does not involve specific objects. Within certain general constraints, GROPER's world model could include any set of objects.

We might instead have used the library of objects for which we are looking to decide how to perform grouping. With this approach, we might look for general characteristics

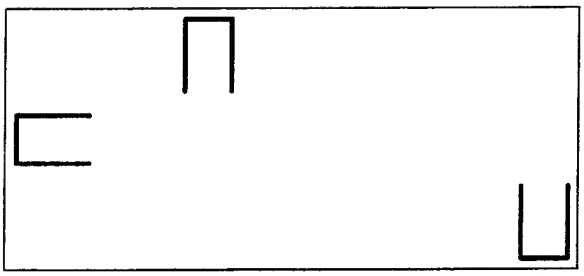

Figure 2.1: GROPER groups together the closest two collections of edges.

of our object library, such as particular angles or other relationships between edges which frequently occur. Turney et. al.[24] and Knoll and Jain[15] have taken this approach to some extent. GROPER does not, for several reasons. First of all, in order to use different libraries, this approach would require us to automatically analyze each particular collection of objects. Some libraries might provide particularly useful characteristics on which to base grouping, while others might not. Secondly, large libraries of objects with flexible parts will require quite general models of the world. With such libraries we can depend less on finding a fortuitous set of characteristics which their objects have in common. We may, nonetheless, still expect to find some generalizations which will apply to almost any library of real objects. For example, we may find that almost any such library will tend to have objects with a large number of parallel edges. However, a third reason for first analyzing a very general model of objects is that the constraints that prove useful for grouping with such a model will probably also apply to more specific sets of objects. We can then add extra constraints suited to a particular domain, such as the tendency of objects to produce many parallel edges.

The general constraints which GROPER uses involve the distance between groups of edges and their relative orientation. For example, GROPER will decide that two groups of edges that are close together are more likely to come from a single object than two groups that are far apart, all other factors being equal. In figure 2.1 GROPER will find the closest pair of groups of edges most likely to come from one object. This principle may seem intuitively obvious, and in fact Gestalt psychologists have noticed that people group together edges on this basis. But this intuition leaves many questions unsettled, such as exactly how the likelihood of edges coming from the same object changes, as the distance between them changes. We can get a sense of these details by analyzing a general model of the world.

Similarly, we can draw some conclusions about the effect of the orientation of two groups

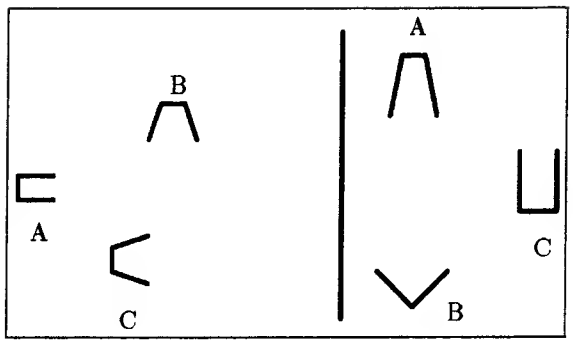

Figure 2.2: In each image, GROPER prefers the combination of groups A and B over the combination of groups A and C, due to the groups' relative orientation.

of edges on the likelihood that they come from a single object. In brief, we find that this likelihood depends on the degree to which two groups of edges point at each other. Figure 2.2 depicts two examples in which GROPER will choose one pair of groups of edges over another because of their orientation. Together, these distance and orientation constraints form the basis for GROPER's grouping system.

2.4 What is Indexing?

In GROPER, indexing takes a group of edges produced by the grouping component and decides which edges from what objects might have produced them. An object might produce a group of image edges if, for some possible position and orientation of the object in the scene, its perimeter aligns with the edges in the image, within some error bounds that reflect the imperfection of the sensing and edge detection. Two further requirements emerge from the way GROPER uses indexing: GROPER can tolerate certain kinds of mistakes more readily than others, and GROPER requires a flexible indexing scheme that will work on any set of input edges.

Verification will correct some mistakes that indexing might make, while it can not correct other types of mistakes. In particular, if indexing produces some matches that really will not work out, verification will discover that, preventing GROPER from making a false positive identification of an object. However, if GROPER's indexing fails to produce a correct match between a group of image edges and an object, then GROPER may lose forever the chance to identify that object. As a consequence of this, indexing must produce all feasible matches between the group of edges used as input and the object models in its library, but it may also produce some matches that are not really feasible.

The edges which GROPER groups together have no definite characteristics. In contrast, Schwartz[22] performs indexing using connected sections of edge, while Lowe[18] groups together edges that have a special relationship, such as parallelism or cotermination. So GROPER needs an approach to indexing that can work for any arbitrary collection of edges.

We use the term indexing to describe this whole process because we use the relationships between the edges in a group to compute an index of a table. Using this index, we look in the table to find the edges of object models that have the same index, and thus the same interrelationship. For example, we might compute the angle between two image edges. Using this angle we can then look in a table to find all pairs of modeled object edges which form that angle. This narrows down the possible pairs of object edges we must consider as potential matches. GROPER computes five different parameters that describe the relationship between two image edges, and uses all of them for indexing. By using indexing to narrow down the number of possible matches, GROPER greatly reduces the amount of computation needed to recognize objects.

2.5 What is Verification?

GROPER uses verification when it already has a good idea of the location of one of the objects in its library, and it wants to make a final decision about whether to accept this hypothesis. Verification makes use of all available evidence. Given a match between image edges and object edges, it estimates the position of the object in the scene. It then looks through all the image edges, not just the ones in the match, to see which ones come close enough to edges of the object model in its suggested position. GROPER then makes a final decision as to whether it has found enough evidence to support the hypothesis in question. It does this, rather arbitrarily, based on whether the edges found account for twenty-five percent of the object's perimeter.

Many computer object recognition systems have made use of similar types of verification, and GROPER does not contribute much that is new in this area. Experiments with GROPER, however, have suggested some shortcomings in this type of verification. In particular, the grouping step was designed so that GROPER would first consider only the combinations of edges most likely to have come from a single object. Verification has no such provision. As a result, verification may result in finding a number of edges that align with a hypothesized object, but that do not all come from a single object. This may result in the incorrect identification of objects. Even when GROPER does correctly identify an object, it frequently decides incorrectly that extra image edges came from that object, and removes them from consideration when it looks for additional objects. This can cause GROPER to miss some objects later.

2.6 Why Grouping?

We can now present a more formal argument for the importance of using grouping in object recognition. This argument again relies on the problems of inaccuracy and complexity that can arise as we deal with more and more complex object recognition tasks. We will argue that grouping overcomes serious problems in the domain in which GROPER operates by contrasting GROPER with an object recognition system that does not use grouping. And we will point out that these problems will become much worse in more complex domains.

If an object recognition system makes use of no principles that tell it which groups of edges to consider first, then it must search through randomly chosen collections of edges in order to find objects. We have implemented a recognition system called SEARCHER which does just that. SEARCHER operates identically to GROPER, only without grouping. Instead, SEARCHER uses a backtracking search.

Although we can take SEARCHER as an indication of how a recognition system will do without any grouping, most recognition systems do not work that way. Instead, they implicitly make use of grouping, or take a quite different approach, as chapter 8 will discuss. But, while SEARCHER is something of a straw man compared to existing recognition systems, the problems it illustrates have mainly been attacked with the use of grouping. GROPER differs from most other recognition systems not in using grouping, but in explicitly analyzing the grouping problem in order to do as much and as effective grouping as possible.

When SEARCHER randomly chooses groups, it usually picks groups of edges that do not all come from the same object. But often these edges still line up with some known object, at some orientation, causing SEARCHER to pursue false leads, and frequently to incorrectly recognize objects not in the scene. Typically the mistakes SEARCHER makes seem ludicrous to a human for one of three reasons. One is that SEARCHER decides an object is on one side of an edge, while a human observer can clearly see that that side is background. SEARCHER, like GROPER, does not start off knowing which side of an edge is figure and which is background. A second is that sometimes in its interpretation SEARCHER uses two different edges that to a person seem to clearly come from two different objects. A third reason for some of SEARCHER's failures is that it may make use of some edges that do seem to come from a single object, without also using some other edges that clearly come from the same object. These difficulties result in SEARCHER making a large number of false identifications when it operates in GROPER's domain. Figure 2.3 gives an example of the objects that SEARCHER finds in an image. Figure 2.4 highlights the three types of mistakes SEARCHER typically makes.

A variety of reasons might explain why other recognition systems do not produce the

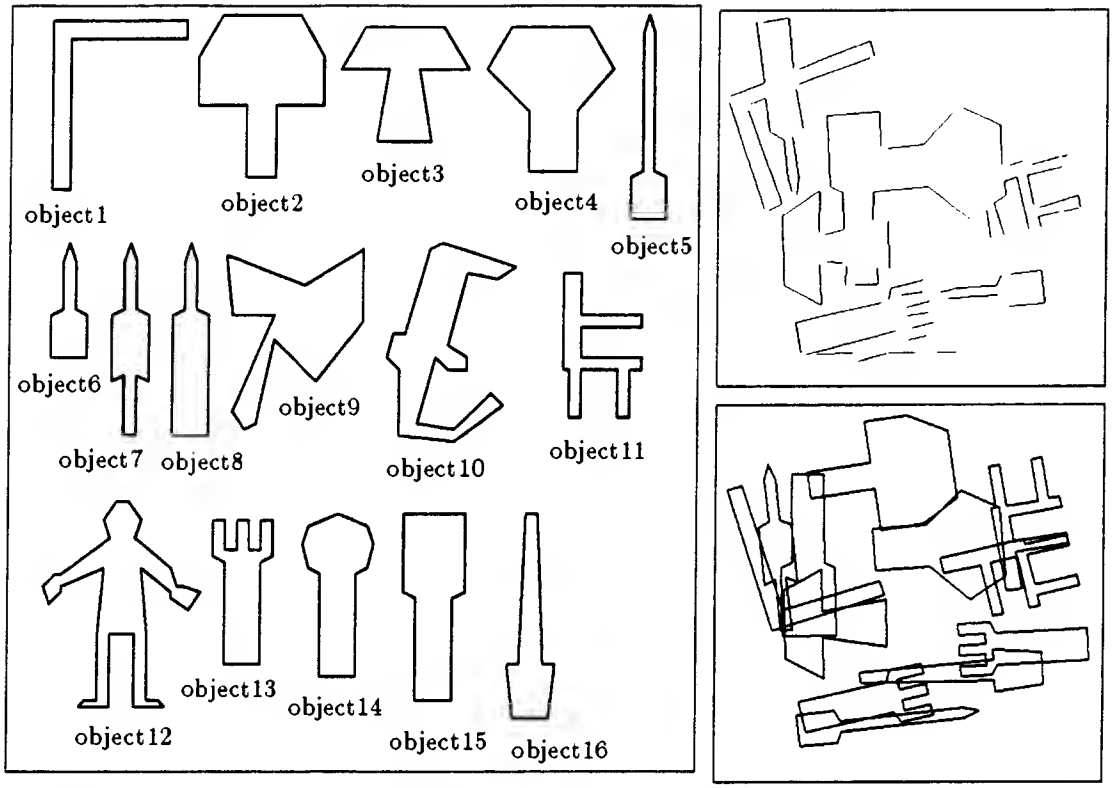

Figure 2.3: The left-most image shows the outlines of sixteen different objects about which SEARCHER knows. The picture in the upper right shows straight lines that approximate the edges in an image containing these objects. The lower right picture shows the objects SEARCHER found, and their location.

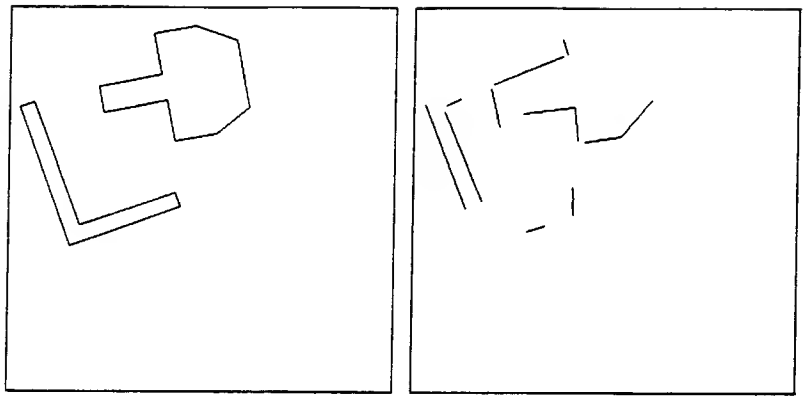

Figure 2.4: On the left are two objects that SEARCHER incorrectly found in the image. On the right, the edges SEARCHER used to recognize each object. SEARCHER found the top-most object due mainly to figure/ground reversals. In both cases, it combines some edges that do not seem to come from the same object, and leaves out some edges that do seem to come from the same object as some of the edges it has chosen.

same inaccuracies that SEARCHER does. First of all, many systems use higher thresholds in deciding when to accept evidence as indicating the presence of an object. Higher thresholds would reduce the number of false positive identifications that SEARCHER makes, but would cause it to miss some objects that GROPER can find. Secondly, as previously noted, these systems often make use of a limited amount of grouping, which can improve accuracy. Thirdly, many systems assume that a preprocessing step can determine the side of an occluding edge on which the occluding object lies. This assumption would reduce the number of errors that SEARCHER makes, too. Fourth, many recognition systems operate in a domain less likely to produce false positive identifications of objects. If a system looks for one object instead of many, there is less chance that an arbitrary collection of edges will match an object it knows about. If a scene contains no objects that even slightly resemble the object for which the system is looking, it will take more of a coincidence to produce a set of edges that together look like that object. And if a system looks for a complex object with many edges, it will take more of a coincidence for unrelated image edges to line up with that object's edges than it would if a system looked for a simple object with fewer edges. All these reasons make it easy to understand the reliability of existing object recognition systems. The fact that SEARCHER looks for a number of simple and quite similar objects, with no ability to tell figure from background, increases its failure rate.

But we can see that even greater problems will emerge when we address domains more complex than that in which GROPER operates. When a system knows about more objects, which can bend or assume any three dimensional orientation at any distance from the

viewer, it becomes much more likely that a random assortment of edges will match some known object in some possible pose. Furthermore, while accurate ways exist of telling figure from ground in some special, industrial settings, we do not yet have a fully general method for performing this task. This growing problem of accuracy provides one of the motivations for the development of methods of grouping that are as general and accurate as possible.

Effective grouping also reduces the amount of work needed to recognize objects. Clearly, the more quickly we find groups of edges that come from a single object, the more quickly we can recognize an object. However, the degree to which grouping speeds up recognition depends on how a recognition system would otherwise structure the search for objects. With the most straight-forward search, using a library of known objects will increase the cost of that search linearly with the number of known objects. By making use of indexing we can reduce that cost, but problems still arise that grouping can help diminish.

We might look at recognition as a search for objects through the space of all sets of image edges matched to sets of edges from an object model. This space is huge, but of course no recognition system needs to explicitly explore it all, since a backtracking search eliminates most possibilities without explicitly considering them. However, since the search proceeds independently for each object in the library, it increases linearly with the size of the library.

By using indexing we can improve on this linearly rising cost. With indexing, we must deal only with the search space of all sets of image edges. For each set of image edges, we perform a single indexing step that tells how many sets of model edges might have produced this set of image edges. The cost of an indexing step may rise with the number of objects in the library, depending on how we perform indexing. A later chapter will discuss this issue in more detail. But in general we can keep this cost low, reducing our search to one through the space of all sets of image edges. It might seem that with indexing, the cost of recognition does not increase at all with the number of objects in the library, except in so far as this effects the cost of a single indexing step, because the number of sets of image edges does not depend on the number of objects in a library.

However, if we make use of a backtracking search, this is not true. A backtracking search benefits us because it takes advantage of the fact that when we find a group of image edges that no set of model edges might have produced, we never need to consider any other set of image edges containing this set as a subset. The more objects we have in the library, the greater the chances that a randomly chosen set of edges will match the edges of some model in the library. This means that we will have to pursue false leads longer before we can backtrack. With only one object in the library, we might primarily have to consider only sets of two or three image edges. These small sets usually would not match the object in the library, unless they were really leading to a correct solution. Our

	Number of Trials	Avg. No. of Edge Combinations Considered	Avg. no. of Objects Found out of nine	Avg. No. of Objects Mistakenly Found in Image
GROPER	5	98.8	7.6	.8
SEARCHER	5	25,984.6	5.2	14

Figure 2.5: A comparison of the number of combinations of edges which SEARCHER and GROPER must explore, as well as the relative effectiveness of the two systems, on a set of test images. Chapter 9 contains the results of other sets of experiments, as well.

search then, would be roughly proportional only to the square or cube of the number of image edges. With a large library, however, sets of two or three edges will usually match some object in the library, causing us to explore the search tree more deeply before we can backtrack. This could make our search space proportional to the fourth or fifth power of the number of image edges, or larger. A backtracking search using indexing will take an amount of time that depends on the chances that a randomly chosen set of image edges matches some object in the library of object models. This will depend not only on the size of that library, but also on whether it contains flexible objects, and on the types of scenes used. But clearly, more complex libraries will result in a much more expensive search for objects.

This discussion indicates that indexing alone will not solve the complexity problems produced by large libraries of objects. Grouping can overcome these problems. Table 2.5 indicates that GROPER explores dramatically fewer sets of image edges than SEARCHER in order to recognize objects. GROPER's grouping component proves effective in ordering the search for objects, while SEARCHER can only search at random. This allows GROPER to consider less than .25% of the possibilities that SEARCHER does. GROPER performs more quickly and accurately than SEARCHER because it more rapidly tries collections of edges that were produced by the objects for which it looks.

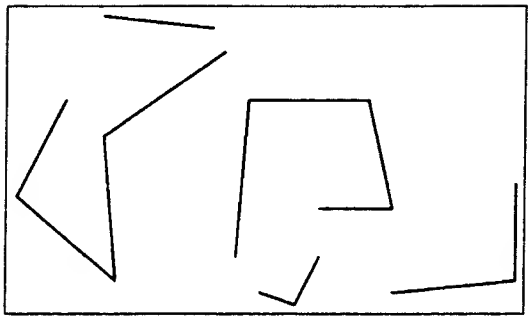

Figure 2.6: Some straight edges that do not resemble anything in particular.

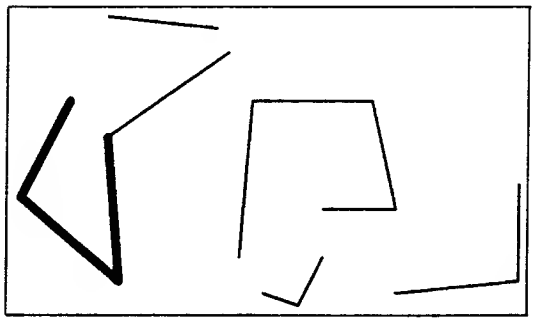

Figure 2.7: Three of those edges seem likely to come from the same thing.

2.7 What Can We Expect from Grouping?

Previous sections have argued for the usefulness of grouping. They have also pointed out that even imperfect grouping can lead to an improvement over a recognition system that relies solely on search. This section will attempt to provide a rough idea of the expectations we should have for a grouping system by looking at the nature of the problem. It will first point out that people appear to be good at grouping. It will then note that the nature of the grouping problem exposes it to some inherent ambiguities that make it unlikely that we can use grouping to completely divide an image into the objects that produced it.

People seem quite good at decomposing natural scenes into their component objects, even when they can not recognize any of those objects. In a scene of unfamiliar objects, a person may recognize the green thing in the corner as a separate entity, without knowing exactly what it is. People can do this even when an image consists only of straight edges. Figure 2.6, for example contains some straight edges. It seems intuitively, at least to the author, that the edges highlighted in figure 2.7 all come from a single object. In any event,

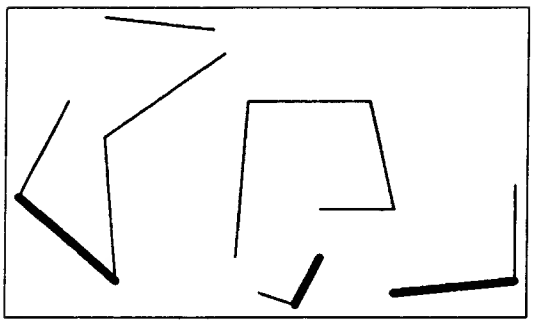

Figure 2.8: These three edges seem less likely to come from the same thing.

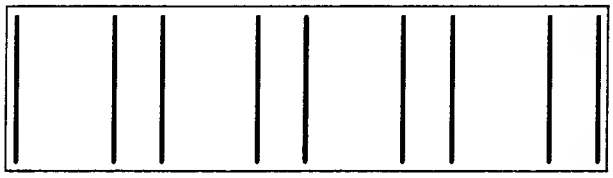

Figure 2.9: People tend to group together the nearby pairs of edges, not the more distant pairs.

certain possibilities seem naturally more likely then others. One can at least easily imagine that the edges highlighted in figure 2.7 come from one object, but one has a harder time seeing the edges shown in figure 2.8 as coming from just one object.

In fact, the gestalt psychologists have noted that people tend to combine edges into groups. In figure 2.9 people tend to form pairs out of the edges close together, while in figure 2.10 people combine the edges that point at each other, even though they are not the closest together. These examples resemble some given by the Gestalt psychologist Wertheimer[26]. As a possible explanation for this phenomena the gestaltists suggested that people group together edges particularly likely to come from the same object. Given that people seem quite good at separating objects in a scene, even before they recognize those

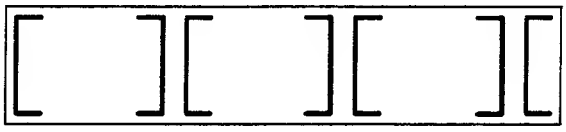

Figure 2.10: In this figure, people see the edges that point at each other as separate groups.

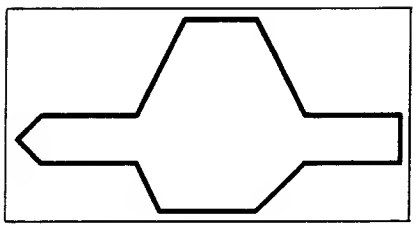

Figure 2.11: Edges like this could easily depict a single object, or two different objects. To decide, we must know what we are looking at.

objects, and that such a separation can make it easier to quickly and accurately recognize objects, it seems plausible that people in fact recognize objects by first performing some grouping step, and then identifying the resulting groups. Of course that grouping step probably does not look exactly like the type of grouping that GROPER does. Chapter 3 will discuss some additional information that human grouping may well make use of, at the least. But even GROPER's somewhat limited grouping can explain the type of grouping people do with figures 2.9 and 2.10, for GROPER picks the same edges that people do as the groups of edges most likely to come from a single object.

The nature of the grouping problem should not, however, lead us to expect a perfect answer, which tells with complete accuracy which parts of an image come from a single object. This holds particularly for grouping that only makes use of the edges in a scene. First of all, we cannot simply make use of connectivity to perform grouping. Edges from the same object will fail to connect because part of the object blends into its background, or another object occludes it. And connected edges will often come from different objects that overlap. Figure 2.11 depicts this inherent ambiguity. An observer cannot tell whether the figure contains one object or two or more, without some model of objects expected in the image. And since the grouping phase, as we define it, makes no use of knowledge of specific objects, grouping alone can not divide such an image completely into its component objects.

So even with grouping, we still expect object recognition to involve some search. A recognition system must explore different possible combinations of edges before it can find the ones that match known objects. Recognition and grouping should interact. A recognition module can indicate which groups of edges match some known objects and so bear further investigation. It can also determine that some groups of edges match no known objects, indicating a need to shift attention elsewhere. At the same time, a grouping system can greatly reduce the work that the recognition system needs to do, and increase its

chances of success.

Chapter 3

A Theory of Grouping

3.1 Introduction

To understand how to form groups from an image we first need to understand the nature of the problem as well as possible. In Marr's terms (Marr[20]) we need a computational theory of grouping. The previous chapter has described the input and output of the grouping process. We now need to understand the regularities in the world that make grouping possible. For example, we might feel at an intuitive level that nearby edges in an image have a greater likelihood of coming from the same object than edges separated by a larger distance. But we must understand why the world would make this true. This understanding will then enable us to make this intuition precise, so that we may answer questions such as how exactly this likelihood varies with distance. This chapter will examine a simplified world model to get a rough idea of what factors we should take into account in performing grouping on images produced by the real world.

Analyzing a simplified world provides only an approximation to a true theory of grouping. Ideally we would like to answer questions such as: "in an image of an average scene how far apart do edges produced by the same object tend to be?", and "how does this compare to the distance separating edges from different objects?". But to answer such questions, we would need to know what an average scene looks like. This would require a complete description of the kinds of objects that exist in the world, as well as an understanding of how often each object tends to appear, and in what orientations. We can not hope to produce such a complete model.

Instead this thesis analyzes a model of a much simpler world. Even the simplified world discussed in this chapter does not lend itself easily to a complete analysis. However, looking at a simplified world develops our intuitions about grouping, leading to hypotheses that we can test with GROPER.

In addition to the difficulty of modeling the real world, another source of difficulty comes from the large number of factors that might influence our assessment of the probable correctness of a group. Suppose we consider digitized images. We can characterize a group by the pixels (dots in the image) that form it. Each pixel in the group has a location and intensity. We wish to find the probability that all the pixels came from the same object in the scene. Any change in the intensity or location of any pixel in the group, or any pixel not in the group, might cause this probability to change. Looked at in this way, a theory of grouping requires us to solve a conditional probability problem with a huge number of variables.

To make this problem tractable, we will make a number of simplifications and idealizations. For example, we only consider the edges that an edge detector finds in the image. Such a decision has advantages and disadvantages. Edges provide a much more compact form for the data. We have fewer variables that might effect the probability we want to derive. But this simplification comes at the cost of ignoring information that might provide important clues to solving the grouping problem.

By creating an uncomplicated model world, and then making some idealizations the following section will reduce the problem of grouping to one of determining the effect of two factors on the likelihood that a set of intensity edges in an image all come from the same object: the distance between edges and their relative orientation. The sections after that will discuss these two factors in detail.

3.2 Simplifications to the Problem

This section will boil down the grouping problem to a more manageable form. It will do this by considering only an uncomplicated world, and by ignoring many potentially useful pieces of information provided by images. The first subsection of this section will present a model of this uncomplicated world, while the second subsection will analyze the problem mathematically, using some simplifying assumptions to reduce it to a problem of understanding the effect of distance and orientation on the likelihood that two groups of edges come from the same object. However, even the more limited problems to which this gives rise seem still to shed light on the factors of which a more complete theory of grouping would need to take account. The end of this chapter will discuss ways to make this easier problem more like the problem presented by the real world.

3.2.1 A Simplified World

The theory of grouping given in this paper requires geometric reasoning. To make this reasoning easier, it limits the type of objects that may appear in a scene. We assume that all objects are flat, lying on a common plane. We also assume that the scene plane maps orthogonally onto the image plane without perspective distortion. This will make it much easier to determine the effect of assumptions about the world on the images the world produces, because the world and the image have such a simple relationship.

GROPER also only makes use of polygonal models of objects. This does not actually simplify geometric reasoning much, but helps mainly to simplify the recognition process that chapter 6 will discuss.

This thesis also assumes that objects do not have too many points of concavity. We will formulate this assumption more precisely when we make use of it. Roughly speaking, it means that we expect to find objects composed of substantial convex parts. One might imagine a hand made up of five completely convex digits, and a convex base. We would not allow a hand made up of many tiny line segments that form a jagged surface full of points of concavity.

We will assume that the world contains randomly oriented objects of random size. Each object in a scene appears in a random spot that does not depend on the location of any other object in the scene. And each object can occur at any size, chosen from a uniform random distribution. In the real world, of course, objects abut or rest on each other, and do not appear in random locations. Furthermore, to simplify the analysis of occlusions in this world, we assume that a section of an object's perimeter can have at most one other object resting on top of it. When we analyze the lengths of occlusions that tend to occur in this world, we will not need to consider the possibility that two different objects can combine to cover a continuous section of an object's perimeter.

We can also make the grouping problem easier by ignoring some information in the image. This allows us to focus on just one piece of the problem.

First of all, this report will consider only a theory of grouping of the occluding edges in the image.

Secondly, we consider only the local rather than the global grouping process. To evaluate the likelihood of a group of edges coming from the same object, we really should consider two kinds of global factors. First of all, the locations of all the other edges in the scene may effect the probable correctness of the group under consideration. If something in the group seems to go well with edges not in the group, this might provide evidence against a particular grouping. Secondly, we should consider all the relationships between edges in the group. And this group could be arbitrarily large, creating a complex network of

relationships.

To avoid these complexities, we will consider only a simpler case. We assume that we have two groups of edges, each known to come from a single object. Furthermore, for each group we know the order of the edges within the object, so that we know the point at which the section of object producing the edges began, and the point at which it ended. We assume that for each edge in a group, we know which side is figure and which is background. This might seem to contradict GROPER's assumption that it does not begin with knowledge of figure and ground. However, GROPER, in using a theory that assumes this knowledge, just allows each edge in the image to create two possibilities, one with the figure on one side of the edge, and one with the figure on the other side. Chapter 5 will discuss the grouping algorithm that does this in more detail.

We want to calculate the probability that all the edges in these two groups come from a single object, and furthermore, that no intervening edges of the object actually appear in the image, for some path between the two groups of edges. This allows us to consider a simpler local problem that will provide a first step in solving the more general global problem. When we use the word "group" in the future, it will refer to a group of edges like this: a group of edges with an order, a beginning and an end.

Thirdly, we do not consider the shapes of groups of edges, only their relative orientation. We can characterize a group of edges by the points where it begins and ends, and the direction of the edges at those points. This characterization throws away everything that happens in between those two points. Using this representation, we can no longer make use of facts such as whether or not two groups of edges contain symmetries. But we have simplified the problem so that now we must only consider the effect of a small number of variables on the acceptability of a group. We could have used a more complex description of a group of edges, or used a simpler one. For example, we might have ignored the direction of the edges that start and end a group. The choice made here is based on the hypothesis that this description provides the minimal amount of information that still allows us to perform effective grouping. Changing the direction of the starting and ending edges affects our intuitions about groups, but changing what happens in between these points does not seem to have as big an effect, unless such changes introduce symmetry and parallelism. Figure 3.1 attempts to gives some examples that develop this intuition, although it remains, at heart, a guess.

To summarize, this simplified world has the following characteristics:

- It contains only edges produced by the perimeters of objects.
- It contains only two dimensional objects.

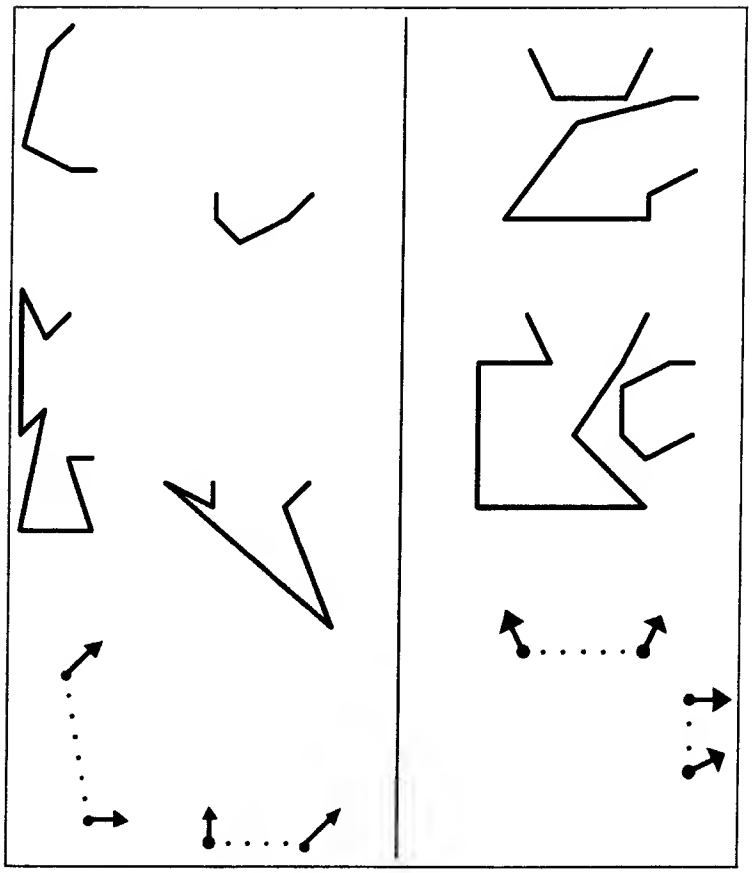

Figure 3.1: The left and right pictures provide separate examples. Each example contains two pairs of groups that have the same characterization, with this characterization depicted on the bottom. Although each of the two pairs of groups look quite different, they seem to "go together" about equally well. That is, our intuitions about whether they come from the same object seem unaffected by the details of the shapes of the edges.

- It contains only polygonal objects. It may contain any set of polygonal objects, as long as they meet the constraints in this list.
- Objects do not have too many points of concavity.
- Objects appear at all different scales, up to some maximum size determined by the size of the image. Within this range, the scale of an object occurs according to a uniform random distribution on the area of the object.
- All objects have a uniformly distributed, random location in the scene. Furthermore, all objects are equally likely to appear.
- No section of an object lies on top of more than one other object.
- Images are formed with orthographic projection.

Furthermore, this analysis considers only local, rather than global relations between groups, using a simplified characterization of each group.

3.2.2 Breaking the Problem into Two Parts

This section will perform some simple probability analysis to make clear exactly what we need to do to solve the problem that remains. To make the problem easier to handle, we will divide it into two parts: the effect of distance and the effect of relative orientation on the likelihood of two groups of edges coming from the same object. We will also find that the assumptions made above still do not allow us to calculate exact probabilities. So instead we will focus on determining the shape of probability distributions. This will allow us to compare the relative likelihood of different pairs of groups coming from single objects. We will see that we can understand what makes one pair of groups more likely than another pair to come from a single object, without determining the absolute probability of either pair coming from a single object.

First of all some notation will make it easier to discuss these concepts. The reader should probably glance at this summary of notation, and then refer to it as the terms appear in the text.

O_1,O_2: We will discuss the problem of evaluating a combination of two groups of edges. We call the object that produced the first group O_1, and the object that produced the second group O_2.

$O_1 = O_2$: This denotes the fact that the two groups of edges come from the same object.

$O_1 \neq O_2$: This indicates that they come from different objects.

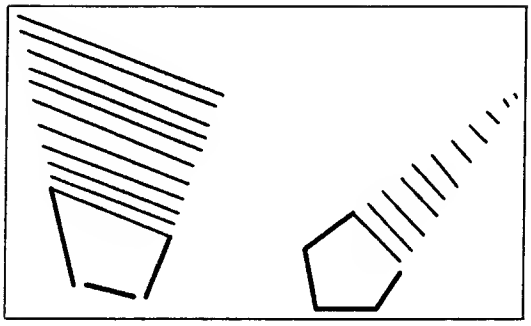

Figure 3.2: Two groups. The shaded area indicates their projection. The projection of the group on the left goes on infinitely.

projection Intuitively, projection refers to the area pointed to by a group of edges. More precisely, consider the following three half-planes derived from a group. First, the line defined by the beginning and end points of the group forms a half-plane that does not include the group. Second and third, the lines defined by the first and last edges in the group form half-planes that include all the edges of the group. We call the intersection of these half-planes the group's projection. Figure 3.2 provides an illustration of projections.

$a_1, a_2, a_3, a_4, l_1, l_2, d, type$: These symbols denote the variables that describe the two groups in question, and their relationship. Figure 3.3 depicts these variables.

- l_1, l_2: Each group has two end points. l_1 indicates the distance along a straight line from the beginning point to the end point of group one. l_2 denotes a similar distance for the second group.
- a_1, a_2, a_3, a_4: We also need two angles to describe each group. One will indicate the angle at which the group's projection spreads out, that is, the angle between its first and last edges. The other tells us the angle between the group's projection and the group itself. a_1 and a_2 will indicate the first of these angles for groups one and two respectively. a_3 and a_4 denote the second angles for groups one and two.
- d: This represents the minimum distance between the ending point of one group and the beginning point of the other. We arbitrarily define the beginning point so that if we follow the group from beginning to end we keep the object that produced the group on the right. We do not need to consider the possibility of an object with a perimeter that joins the beginning of one group to the beginning

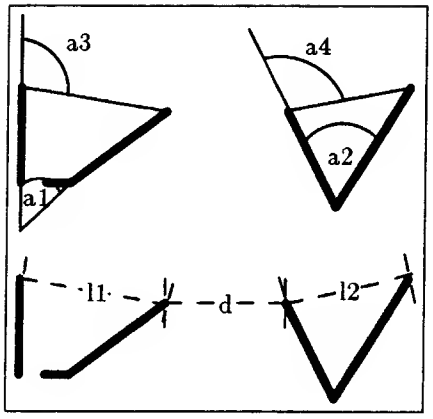

Figure 3.3: Seven of the variables that describe the relationship between two groups. An eighth variable tells us the groups have a $type_2$ relationship.

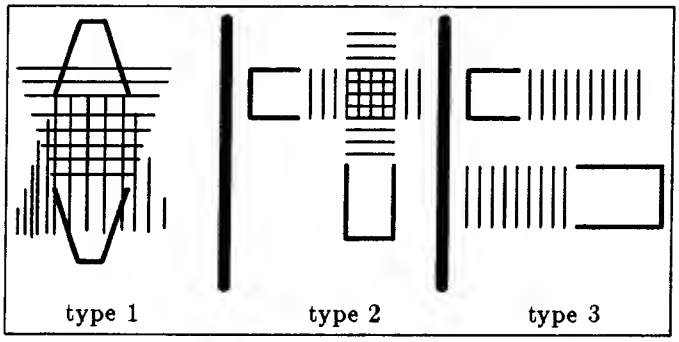

Figure 3.4: The three types of orientations that groups may have. The shaded areas indicate the projections of the groups.

of another, because such a connection would cause figure and background to reverse.

$type_1, type_2, type_3$: We have used seven variables to capture (almost) seven of the nine degrees of freedom that describe two groups and their relationship. The remaining two variables describe the relative orientation of the two groups. The *type* of the two groups divides this orientation into three classes.

$type_1$: *type* has this value when the two groups could come from a single convex section of an object. This happens when all the edges in each group fall inside the other group's projection.

$type_2$: This value indicates that the two groups could come from adjacent convex

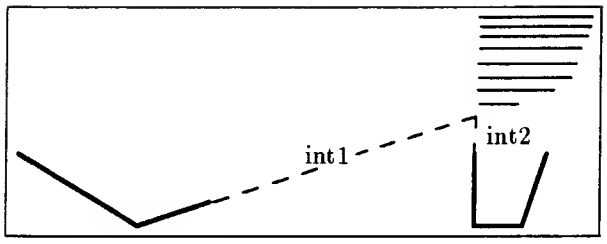

Figure 3.5: int_1 and int_2, the distance to the intersection of the projections of two groups. The shaded area indicates where the projections intersect.

sections of the same object. This occurs when the two projections intersect.

$type_3$: *type* has this value when it does not have the value $type_1$ or $type_2$.

Sometimes we will abbreviate the situation $type = type_1$ by just saying $type_1$. Context should make this clear. Figure 3.4 provides examples of the three types of orientations.

int_1, int_2: These variables apply only if $type = type_2$. In that case, the projections of the two groups intersect. int_1 is the minimum distance to this area of intersection from the first or last point in the first group. Suppose these two groups really do come from adjacent convex sections of the same object. Then int_1 represents the minimum amount of perimeter that is part of the same convex section as the edges in the first group, but that did not show up in the image. We define int_2 similarly for the second group. Figure 3.5 provides examples of intersection distances.

data, *rest of data*: We will abbreviate $d, a_1, a_2, a_3, a_4, l_1, l_2, type, int_1, int_2$ with *data*. We will refer to all the *data* except for some obviously excluded variables with *rest of data*. Context should make clear to which variables *rest of data* refers.

We can now state formally what we would like to compute:

$$P(O_1 = O_2 | d, a_1, a_2, a_3, a_4, l_1, l_2, type, int_1, int_2)$$

That is, we want to find the conditional probability that two groups of edges come from the same object, given values for all the above variables. Since these variables do not fully specify the relationship between the two groups, this formulation of the problem makes a simplification by ignoring some information available in the relative orientations of the groups.

Using Bayes' theorem we can transform this problem into an equivalent one:

$$P(O_1 = O_2|data) = \frac{P(data|O_1 = O_2) * P(O_1 = O_2)}{P(data)}$$

Since

$$P(data) = P(data|O_1 = O_2) * P(O_1 = O_2) + P(data|O_1 \neq O_2) * P(O_1 \neq O_2)$$

we find that:

$$P(O_1 = O_2|data) = \frac{P(data|O_1 = O_2) * P(O_1 = O_2)}{P(data|O_1 = O_2) * P(O_1 = O_2) + P(data|O_1 \neq O_2) * P(O_1 \neq O_2)}$$

$P(O_1 = O_2)$ and $P(O_1 \neq O_2)$ refer to the probability of two groups of edges coming from the same object in the absence of any information about the two groups or their relationship. This will depend on the number of objects we expect to find in a scene, and on how many groups of edges we expect each object to produce. However, we do not need this information to compare the relative merits of two different combinations of groups in an image. To see this, notice that:

$$\frac{1}{P(O_1 = O_2|data)} = 1 + \left(\frac{P(data|O_1 \neq O_2)}{P(data|O_1 = O_2)} * \frac{P(O_1 \neq O_2)}{P(O_1 = O_2)}\right)$$

Suppose we have two sets of data, *data* and *data'*, for two different combinations of groups. $P(O_1 \neq O_2)$ and $P(O_1 = O_2)$ are the same for any combination of groups in an image because they do not refer to the parameters that describe those groups. So, if:

$$\frac{P(data|O_1 \neq O_2)}{P(data|O_1 = O_2)} > \frac{P(data'|O_1' \neq O_2')}{P(data'|O_1' = O_2')}$$

then:

$$P(O_1 = O_2|data) < P(O_1' = O_2'|data')$$

So in order to pick the most likely of a number of possible groupings, we need only be able to calculate:

$$\frac{P(data|O_1 \neq O_2)}{P(data|O_1 = O_2)}$$

We now need to determine how to separate *data* into component parts, in order to analyze the effects of individual variables on the total probability we need to compute. Notice that:

$$P(data|O_1 = O_2) = P(d|O_1 = O_2) * P(rest\ of\ data|d, O_1 = O_2)$$

This isolates the effect of d, but now we must look at $P(rest\ of\ data|d, O_1 = O_2)$.

$$P(rest\ of\ data|d, O_1 = O_2) =$$

$$P(l_1, l_2, a_1, a_2, a_3, a_4 | d, O_1 = O_2) * P(type, int_1, int_2 | l_1, l_2, a_1, a_2, a_3, a_4, d, O_1 = O_2)$$

We are going to remove the first term in this product by assuming that:

$$P(l_1, l_2, a_1, a_2, a_3, a_4 | d, O_1 = O_2) = P(l_1, l_2, a_1, a_2, a_3, a_4 | O_1 = O_2)$$

and

$$P(l_1, l_2, a_1, a_2, a_3, a_4 | d, O_1 \neq O_2) = P(l_1, l_2, a_1, a_2, a_3, a_4 | O_1 \neq O_2)$$

and

$$P(l_1, l_2, a_1, a_2, a_3, a_4 | O_1 = O_2) = P(l_1, l_2, a_1, a_2, a_3, a_4 | O_1 \neq O_2)$$

These assumptions are not generally true, and so involve a further simplification of the problem.

What do these assumptions mean? The first two state that the variables that describe the shape of two group are independent of the distance between them. If the groups come from different objects, as in the second assumption, then this seems to follow from the earlier assumption that our world contains only randomly oriented objects. However, if O_1 and O_2 overlap, then these variables may not be independent. A single cause, the overlap, may help determine the way both groups begin and end. If the groups come from the same object (assumption one) this implies that the way that two groups of edges in the same object start and finish does not depend on the distance between them. This may not be true in general if, for example, objects have patterns that repeat in only spatially localized areas. The third assumption implies that the way we expect groups to start and end does not depend on whether or not they come from the same object. This may not be true in general for similar reasons. Intuitively, these assumptions mean that we are only dealing with a world that contains random, unstructured objects. By random I mean objects for which knowing about one part of the object does not tell us about other parts of the object.

Why make these particular assumptions, given that the world may violate them? These assumptions arise from the intuition that while the shape of a group may tell us a good deal about the shapes of nearby groups in the same object, the way a group begins and ends *will not* tell us much about the way nearby groups begin and end. We should deal with this type of information only when we have an approach to grouping that tries to take into account the complete shapes of different groups of edges. We make a tactical decision to ignore this kind of information in the hope that we will not go too far wrong, and that we will now be able to focus on more crucial factors.

Given the three assumptions above, and the equation that stated:

$$\frac{1}{P(O_1 = O_2)} = 1 + \left(\frac{P(data | O_1 \neq O_2)}{P(data | O_1 = O_2)} * \frac{P(O_1 \neq O_2)}{P(O_1 = O_2)} \right)$$

we can see that $P(l_1, l_2, a_1, a_2, a_3, a_4 | d, O_1 = O_2)$ and $P(l_1, l_2, a_1, a_2, a_3, a_4 | d, O_1 \neq O_2)$ cancel each other. That is:

$$\frac{1}{P(O_1 = O_2)} \propto 1 + (\frac{P(d|O_1 \neq O_2)}{P(d|O_1 = O_2)}) * (\frac{P(type, int_1, int_2 | rest\ of\ data, O_1 \neq O_2)}{P(type, int_1, int_2 | rest\ of\ data, O_1 = O_2)})$$

The next two sections will analyze the distance part, and the orientation part of this equation.

For convenience, we will list the assumptions relied on in this section that we did not set out in the previous section:

- We only attempt to discover the relative merits of different combinations of groups, not the absolute probability of a group's correctness.
- We ignore any information about a group's orientation not captured in the *type*, int_1, or int_2 variables.
- We ignore the dependence between the angles and lengths that describe two groups and the distance separating the groups.
- We do not analyze the dependence between the way groups start and end, and whether they come from the same object.

3.3 The Effect of Distance

This section will provide estimates of the distributions of $P(d|O_1 = O_2)$ and $P(d|O_1 \neq O_2)$. It will argue that if groups come from the same object, we expect shorter rather than longer distances to separate them. If they come from different objects, we expect the reverse.

3.3.1 When Groups Come from the Same Object

To find $P(d|O_1 = O_2)$ we will ask what causes a section of an object's perimeter to fail to produce an edge in the image. This is not the only cause of some distance separating two groups. There may be another group of image edges that accounts for some of the object perimeter connecting the two groups. However, we focus on the possibility that two groups not only come from the same object, but come from nearby parts of that object because we expect such situations to produce the best groups, which we will want to use first for recognition. We assume that if two groups came from widely separated parts of the same object, the data will prove a less reliable indicator of whether the two groups came from the same object.

Two different causes can prevent a section of an object's perimeter from creating an edge in an image: occlusion and blending into the background. The following paragraphs will argue that these methods tend to create shorter, rather than longer sections of missing perimeter.

Occlusion

We begin by discussing the distribution of the lengths of occluded sections of objects. That is, suppose, given the simplified world we are analyzing, we generate a random occlusion between two objects. We would like to know the probability distribution of the distance separating the beginning and the end of the occluded section of perimeter. This will tell us the distribution of some of the distances that separate different groups that come from the same object. We will proceed by first pointing out a correspondence between the distribution of lengths of occluded object, and the distribution of distances between points within objects. We will first cover a more limited domain, that of completely convex objects, and then discuss how to extend the result to general polygonal objects with some points of concavity. Finally, we will discuss the bearing this result has on the other ways that sections of perimeter fail to produce image edges.

In a two dimensional world, occlusion occurs when one object lies on top of another, blocking part of its perimeter. When one object blocks a contiguous section of another's perimeter it means that the two objects' perimeters overlap at two points, at the beginning and end of this missing section of perimeter. Furthermore, the same distance separates the two points on each object.

It is difficult to determine the likelihood that two randomly oriented objects will produce an occlusion with a length in a given range. But we will argue that this distribution is related to the distribution of the distance separating two randomly chosen points on the perimeter of each object.

To find the real distribution, we could divide each object's perimeter into short sections. Call these sections $(s_1, s_2, ...s_n)$ and $(t_1, t_2, ...t_n)$. To find the probability that an occlusion of length d to $d + \delta$ will occur, we can just sum over all i, j, k, l the probability that an occlusion of that length could arise with sections s_i and s_j overlapping t_k and t_l. This probability will depend on how much of s_i is the appropriate distance from section s_j, and how much of t_k is the right distance from t_l. So, as we make the sections arbitrarily small, the probability distribution will depend on how many pairs of tiny sections on the first object's perimeter are between d and $d + \delta$ apart, and how many on the second object are that distance apart. We can find this out by examining the probability distribution of the distance separating two randomly chosen points from each object.

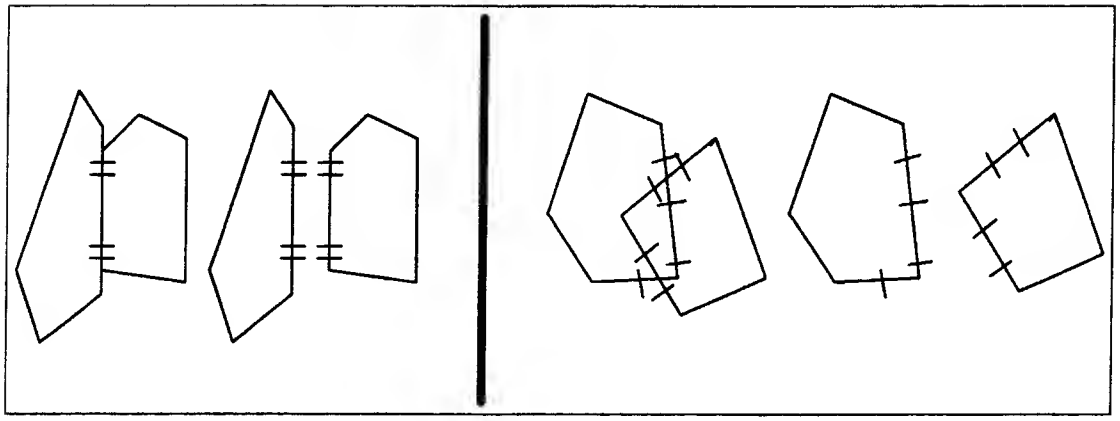

Figure 3.6: Each side of the above illustration shows an occlusion. Each polygon has two sections of its perimeter marked, and each section of perimeter overlaps a section from another object. The objects on the left must align perfectly for an occlusion to occur, while the objects on the right have some leeway in their orientation.

However, the likelihood of two pairs of sections of perimeter, one from each object, producing an occlusion, does not just depend on the distance between the sections, it also depends on the angle between them. Figure 3.6 shows two examples. On the left, we see two objects with two sections of perimeter marked on each object. Because the sections of perimeter are parallel in each object, there is a measure zero probability that an occlusion will cause these four sections of perimeter to overlap. On the right, we see an example of four sections of perimeter that really may produce an occlusion. We will ignore this complication, however, and assume that we may reasonably approximate the distribution of occlusions produced by two objects by assuming that a one-to-one mapping exists between each occlusion of distance d and each pair of points on each object separated by the distance d. We can find this distribution by multiplying the distribution of the distance that separates two randomly chosen points from the first object by a similar distribution for the second object, and then normalizing the resulting distribution so that it has an integral of one.

We will analyze the probability distribution of the distance between randomly chosen points on a single object in order to show that short occlusions will occur more often than long ones. This probability distribution will vary from object to object. So we would like to get an idea of the worst possible distribution for proving our hypothesis. The object that produces the longest occlusions will be the object with the longest distances separating randomly chosen points; we want to analyze that object.

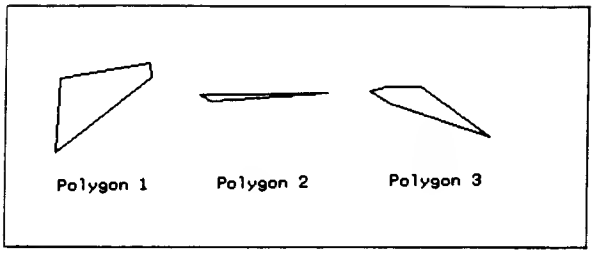

Figure 3.7: Three randomly chosen, convex polygons.

Since we assume that any object appears at randomly chosen scales, we do not care about the absolute distance between two points on an object; that distance will depend on the scale we choose in any event. So, in comparing different objects, we normalize this distance based on the size of the object. To do this we take each object at a scale for which a distance of one unit separates the two points farthest apart.

In this report we conjecture that using this standard of comparison, a circle will provide the worst possible case for our hypothesis. That is, for a circle, the distance between randomly chosen points, divided by the maximum possible distance between two points, will tend to be highest. Intuitively this makes some sense because, for a given perimeter, a circle minimizes diameter while placing each point as far as possible from the others. Figures 3.7 and 3.8 show some support for this. Figure 3.7 shows some randomly chosen convex polygons. Appendix A explains how we construct them. Figure 3.8 shows that pairs of randomly chosen points tend to be much further apart for the circle than for any of the polygons.

We claim that a circle provides the worst case distance distribution in the following sense: that for any distance, d, the probability of randomly choosing two points on the circle less than a distance d apart will be less than it will for any other convex object, providing we normalize each object based on the maximum distance separating any two points. This thesis does not offer a proof of this conjecture, but experiments do support it. Of 400 randomly chosen convex objects, all produce probability distributions for which this conjecture holds.

So suppose circles do offer a worst case for us to consider. This means that a universe of circles will tend to produce longer occlusions than any other universe of objects. But we still assume that circles appear at all possible scales. When we average the circle's distribution over all scales we get a monotonically decreasing distribution. This assumes we weight the averages of the distributions with the perimeter of the circle, as circles with longer perimeters can produce more possible occlusions. This averaged distribution means that if we randomly select two points from a random sized circle, we are more likely to get

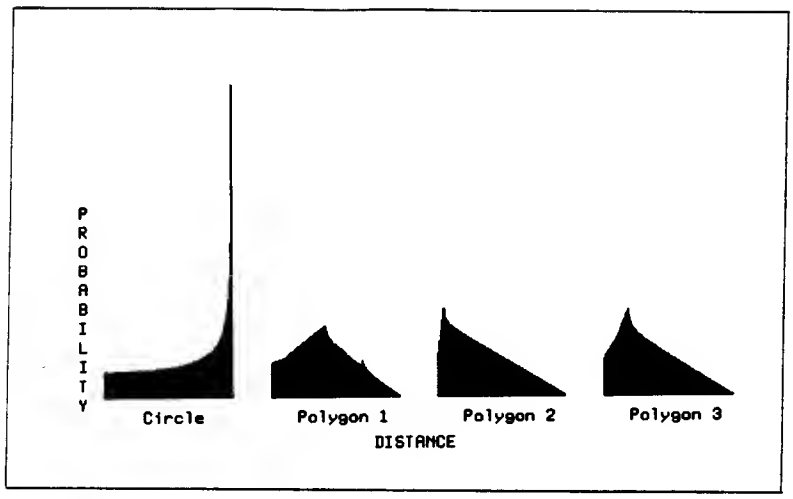

Figure 3.8: The distribution of a variable obtained by randomly selecting two points from the perimeter of a shape, and measuring the distance that separates them. We normalized the circle and the polygons so they have the same diameter.

points separated by shorter distances than longer ones.

Once we have the distribution of distances between pairs of points for a particular library of objects, we can determine the distribution of occlusions for that library by taking the normalized square of the initial distribution. The distribution of lengths of occlusion is just the distribution of the distance separating two randomly chosen pairs of points from the same object, when the same distance separates those points. This distribution corresponds to the distribution achieved through the following process: randomly select two points from a randomly chosen object; then randomly select two more points from another randomly chosen object; if the difference in the two distances between the two pairs is less than δd, we have a random variable; otherwise, start over. As δd goes to zero we get the distribution of lengths of occlusions. Clearly, for any distance d, the chances of choosing two sets of points with distances in the range $(d - \delta d, d + \delta d)$ is just the square of the chances of picking one pair of points separated by a distance in that range. So the distribution of the distances separating two pairs of points is just the square of the distribution for single pairs of points randomly chosen from our library. Of course, after squaring the distribution, we must normalize it so that it has an integral of 1. As we mentioned, this analysis assumes that each occlusion corresponds to a pair of two points on each object separated by an equal distance. This ignores the additional effect of the two angles between the points.

We now have a worst case distribution for the lengths of occlusions produced by any library of convex objects: the distribution produced by a library containing just a circle. Figure 3.9 shows this distribution, as well as the distribution of occlusions we expect when

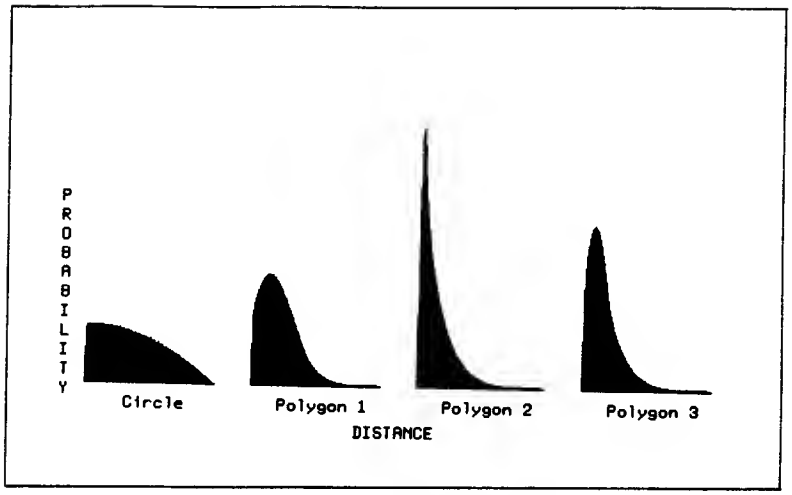

Figure 3.9: The square of the distribution obtained by taking a shape at a random scale, and measuring the distance between randomly chosen points. The distribution is calculated analytically, not experimentally.

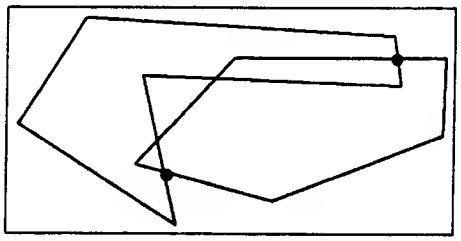

Figure 3.10: Not every set of two pairs of points on two concave objects corresponds to a single occlusion.

our library contains each of the three randomly chosen polygons.

If objects have concavities we no longer have a one-to-one mapping between occlusions and pairs of pairs of points separated by equal distances. With concave objects, in between the points where the two objects overlap, one object's perimeter may dip back inside the others (see figure 3.10). We will not discuss this issue much, but will just point out one reason why we might expect objects with concavities to produce even shorter occlusions.

Consider the case where we have two lengths of perimeter, P_1 and P_2. P_1 begins at point p_{11} and ends at p_{12}. P_2 begins and ends at p_{21} and p_{22}. First of all, it seems obvious that the longer a section of perimeter, the more likely it is to contain a concavity, because we only consider objects with relatively few points of concavity. And unless either P_1 or P_2 contain a concavity, then the set of points p_{11}, p_{12}, p_{21}, and p_{22} will correspond to an

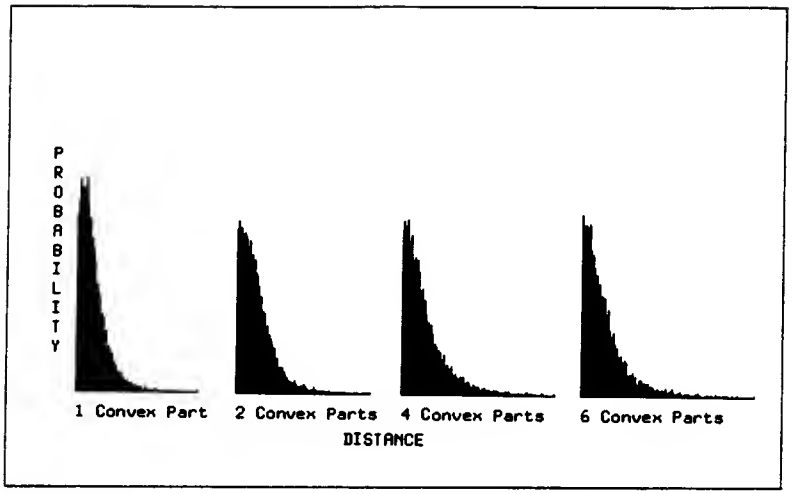

Figure 3.11: The lengths of occlusions that occur from randomly intersecting randomly chosen objects.

occlusion. If they do contain a concavity these points may or may not correspond to an occlusion. Provided we assume that the longer the distance between the end points of a section of perimeter, the longer the section of perimeter is likely to be, the likelihood of a concavity falling inside a section of perimeter then tends to favor short rather than long occlusions.

The other question we would have to consider for full analysis of this problem is whether longer rather than shorter sections of perimeter that do contain concavities have a greater chance of arcing back inside each others boundaries like the one in figure 3.10.

We can test these theoretical arguments empirically. To do so, we have constructed random objects, as described in Appendix A, and used them to form random occlusions. We fabricated convex objects, objects with two convex parts, with four convex parts, and with six convex parts. Figure 3.7 shows examples of random convex objects. Figure 3.11 shows the distribution of lengths of occlusion that these objects produce. We find that long occlusions do occur more rarely than short ones.

This section has not presented a proof that short occlusions occur more often than long ones. Rather, an analysis of the problem has suggested a reasonable conjecture that we can support with empirical evidence.

Blending into the Background

A section of an object's perimeter may also fail to create an edge when it blends into its background, that is, when the part of the scene behind the object has the same reflectance

as the object itself. This section will discuss this process, pointing out that it bears a close resemblance to the process of occlusion. First, we will make use of a simple model of objects' reflectances. For this model, the length of missing perimeter from blending into the background will have the same distribution as the missing lengths due to occlusions. We will then point out that a more realistic model of reflectances will tend to produce even shorter sections of missing perimeter.

First let us assume a quite simple model of object reflectances, which has three properties. One, the scene has a single background plane that has a uniform reflectance. Furthermore, no object in the scene has the same reflectance as this background plane. Think of a scene in which many dark objects lie on a white table. Second, every object has a uniform level of reflectance; the intensity an object produces in an image will not vary throughout that object. Third, each part of the scene has the same level of illumination. These three assumptions imply that an object blends into its background only when it rests on top of another object with the same reflectance. No object blends into the scene's background plane.

In such a world we find a correspondence between lengths of perimeter missing from blending into the background and missing from occlusion. Whenever one object lies on top of another, occluding it, if the two objects have the same reflectance the object on top will blend into the object on the bottom for exactly the length of the occlusion. Not every occlusion also causes blending in, the two objects may have different reflectances. But since the scene contains only randomly oriented objects, we may assume that the reflectance of two objects bears no relation to the length of the occlusion they produce. So, some random subset of the occlusions in a scene will also produce missing lengths of perimeter due to the object on top blending into the object on the bottom. And this missing perimeter will have the same distance between its start and end as the occlusion did. Therefore, the distance between the start and end of sections of an object that blend into the background will have the same distribution as the distance produced by occlusions.

We simplified the previous section by not allowing two occlusions to join together to form one long occlusion. This same restriction prevents two separate events of blending into the background to join together.

Now we will briefly discuss what happens with a more realistic model of the world. First of all, in a more realistic model, objects may occasionally have the same reflectance as the background plane. However, we can still assume that this happens infrequently, and plays only a minimal role in the total effect that we are studying. Secondly, in the new model the reflectance of an object may vary. These variations in reflectance will produce even shorter sections of missing perimeter than the simpler model.

Even in this more complicated model, each section of missing perimeter will usually

stem from one object lying on top of another object with the same reflectance. But now, the section of missing perimeter may be shorter than the occlusion. Where one object lies on top of another, the two objects may have the same reflectance for some short space of time. But the reflectances of the two objects will change in independent ways. This means that the two objects may have the same reflectance for some space, but at any point there is a chance that the reflectance of one of the objects will change. Only through an unlikely coincidence will the reflectance of the other object change in the same way. So, at worst, the two objects might have a constant reflectance throughout the length of the occlusion, creating a missing section of perimeter equal in length to the length of the occlusion. This more complex model alters things only by allowing one of the object's reflectances to change somewhere in the middle of the overlap, which will usually result in a shorter section of missing perimeter than that produced by the occlusion. Therefore the distribution of sections of perimeter missing from blending into the background will tend to produce even shorter sections of missing perimeter than will occlusions.

When Nothing Gets in the Way

The previous two sections discussed the likelihood of events occurring that will prevent part of an object's perimeter from making an edge in the image. Neither discussion indicates the probability that a distance of 0 will separate two groups of edges, because nothing has caused their perimeter to fail to appear. Fortunately, since we only need to calculate:

$$\frac{P(d|O_1 = O_2)}{P(d|O_1 \neq O_2)}$$

we do not need to determine either the numerator or denominator separately. The forthcoming discussion will show on what this ratio depends.

3.3.2 When Groups Come from Different Objects

This section will analyze the problem of determining the chances of a given distance separating edges that come from different objects. We will do this in stages. First we will do it for a somewhat simplified version of the problem. We will then point out flaws in this analysis, and get a more accurate approximation of the answer.

A Simple Analysis

Each group of edges has a beginning and an end. If the sections of objects these groups come from are randomly oriented in the image, we can assume (falsely as we will see) that the beginning and end of each group has a random location. The distance between the

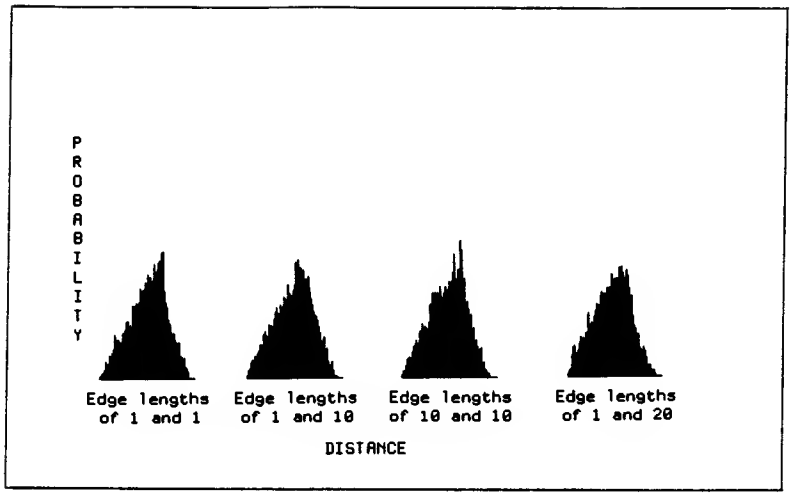

Figure 3.12: The distribution of distances between randomly oriented groups. This was obtained by randomly orienting two line segments in a square image, and finding the minimum distance from the start of one to the end of the other. The likelihood rises steadily, and then begins to drop as the size of the image begins to restrict the distance that can separate the two edges.

beginning and end of a group will of course constrain their location. So this situation is just like randomly placing two line segments in the image.

We can now find the distance between the two groups, as we have defined this distance, by taking the minimum of the distance between the beginning of each group, and the ending of the other. Randomly locating two line segments is like randomly locating the beginning of one and the ending of another, and then randomly locating the remaining two points, subject to the constraint provided by the known lengths of the segments. If we call the beginning of the two groups p_{11} and p_{21}, and the ending points p_{12} and p_{22}, we find we need to compute the distribution of:

$$P(||p_{11}, p_{22}|| = d_0) * P(||p_{21}, p_{12}|| >= d_0) + P(||p_{12}, p_{21}|| = d_0) * P(||p_{11}, p_{22}|| > d_0)$$

Figure 3.12 shows the distribution this process produces for several sample values of lengths separating the beginning and ending of the two groups. Analytically we can see that for small groups the probability will increase linearly with the distance between the two groups. We can see this because the points in the plane within a given distance away from a point, fall within a circle. The area of this circle increases with the square of its diameter. The derivative of this, i.e. the rate of change of the area, increases linearly with the distance.

A Refined Analysis

The above analysis made the unwarranted assumption that the position of the beginning and end points of one group of edges is independent of the beginning and end points of the other group. But suppose that the two objects overlap, covering up part of one of these two groups of edges. One of the groups of edges might have to end at the point of the occlusion. Let us consider a simple case. If the two objects have the same reflectance, the perimeter of the object on top will not create any edges where it lies over the bottom object. The two sections of perimeter will co-terminate, creating a single curve, with a concavity where they meet. (Two objects form a concavity at a point of occlusion except in the case of an extremely unlikely alignment of the objects). A distance of 0 will separate the two groups of convex edges formed in the image. To create a more realistic analysis, we must consider what happens if the two groups of edges had their end points determined by an occlusion involving both of them.

Two factors will influence the impact of such an occlusion. First of all, we must consider the probability of such an occlusion occurring. Then we must consider the possible distances between the groups when such an occlusion has occurred.

If no such occlusion occurs, we can use our previously developed analysis. So, we want to combine that analysis with an analysis of what happens when an occlusion occurs, based on the probability of an occlusion.

$$P(d|O_1 \neq O_2)$$

$$= P(d|O_1 \neq O_2, \textit{groups do not overlap}) * P(\textit{groups do not overlap}|O_1 \neq O_2)$$

$$+P(d|O_1 \neq O_2, \textit{groups overlap}) * P(\textit{groups overlap}|O_1 \neq O_2)$$

We do not develop the probability of such an occlusion occurring.

In the case where the two objects have the same reflectance, an occlusion of the two groups has a simple interpretation. One of the groups ends where the other begins. Thus, distance equals zero, unless something else occludes this point. This means that in the case of an occlusion, we get essentially the same distribution of distances that we get with two groups from adjacent convex sections of the same object. If nothing occludes this connection, we get a distance of zero between the two groups. If something does occlude them, the distance between the two groups equals the length of the occlusion. This also provides an answer to a question raised earlier.

$$\frac{P(d = 0|O_1 = O_2)}{P(d = 0|O_1 \neq O_2)}$$

is just one divided by the probability of two groups of edges from different objects overlapping.

So, the distribution of distances produced by groups coming from different objects equals a linear combination of two curves. The first curve also describes the distribution of distances when the groups come from the same object. This curve applies when the groups overlap. The second curve describes the distribution of the minimum distance between the end points of two randomly oriented line segments. The way we combine these two curves depends on the chances of two groups from different objects occluding each other.

A Refined Refined Analysis

Unfortunately, this analysis is not quite right either. We did not consider all the possibilities when we said that either the two groups have independently located end points, or else they intersect. For example, if the two groups fall close to each other, a third object might occlude them both. A complete analysis would require considering factors that might affect the location of the end points of the groups in a non-independent way, even when the groups do not intersect. We will not consider these cases, however. We assume that the distribution of distances separating two groups in this situation will not differ much from the distribution when the groups have independent locations.

3.3.3 Summary

Although incomplete, our analysis suggests that the likelihood of a distance separating two groups that come from the same object will tend to decrease with the distance, with the distribution a circle produces as the worst case. The distribution when the groups come from different objects will approximate a linear combination of this curve and another that increases linearly with the distance. For larger distances, the second curve will dominate. For smaller distances, the first curve will. So for smaller distances, $\frac{P(d|O_1=O_2)}{P(d|O_1\neq O_2)}$ will become the inverse of the probability that two groups of edges from different objects intersect. For larger distances the value of this ratio will decrease rapidly.

3.4 The Effect of Orientation

This section will discuss the impact of different factors on the relative likelihood of different types of orientation occurring. Dividing the possible relative orientations of groups into three types provides only a coarse description of them. This description proves useful nonetheless because types corresponds to something real about objects. The type classification of two groups tells us whether they could have come from the same convex section of an object, from adjacent convex sections, or only from non-adjacent sections. This makes

it easier to calculate the probability of different types occurring if the groups come from the same object.

This section will leave some questions unanswered about the exact probability of certain events occurring. It will try instead to understand the general shape of the distributions in question, as well as which variables in the data will influence this shape, and which variables we can hope to ignore.

3.4.1 When Groups Come from the Same Object

This section will consider the probability of each of the three possible types occurring when two groups come from a single object. Recall that the type of orientation of two groups indicates whether they could have come from the same, adjacent, or only non-adjacent convex parts of an object. The discussion of all of the types will follow similar lines. For each type, we will divide the probability into three parts. What if the two groups actually come from the same convex section of an object? What if they come from adjacent sections? What if they do not? This will lead to questions like, what is the probability that groups coming from adjacent convex sections of an object will appear to have come from adjacent convex sections? We will need to discuss the following nine problems (abbreviating "groups actually come from same section" with "*same*", "groups come from adjacent sections" with "*adj*", "groups come from non-adjacent sections" with "*notadj*"):

$$P(type_1|same, rest\ of\ data, O_1 = O_2)$$

$$P(type_1|adj, rest\ of\ data, O_1 = O_2)$$

$$P(type_1|notadj, rest\ of\ data, O_1 = O_2)$$

$$P(type_2|same, rest\ of\ data, O_1 = O_2)$$

$$P(type_2|adj, rest\ of\ data, O_1 = O_2)$$

$$P(type_2|notadj, rest\ of\ data, O_1 = O_2)$$

$$P(type_3|same, rest\ of\ data, O_1 = O_2)$$

$$P(type_3|adj, rest\ of\ data, O_1 = O_2)$$

$$P(type_3|notadj, rest\ of\ data, O_1 = O_2)$$

We choose to decompose the problem in this way because many of these probabilities have simple answers. For example, $P(type_3|adj, data, O_1 = O_2)$ equals 0, since if two groups come from adjacent convex sections of an object, they can not look as if they could not

possibly come from the same or adjacent convex sections. Two other probabilities also equal zero, while $P(type_1|same, data, O_1 = O_2)$ equals one.

The probabilities discussed here will all depend on the probabilities of the sections of objects from which the two groups come having various kinds of relationships. For example, to calculate some probabilities, we would need to know the chances of two randomly chosen groups of edges in the image coming from the same convex section of an object, based on some of the data and on previous expectations. We see this from the equations :

$$P(type_1|rest\ of\ data, O_1 = O_2)$$

$$= P(type_1, same|rest\ of\ data, O_1 = O_2) + P(type_1, adj|rest\ of\ data, O_1 = O_2)$$

$$+P(type_1, notadj|rest\ of\ data, O_1 = O_2)$$

$$= P(type_1|same, rest\ of\ data, O_1 = O_2) * P(same|rest\ of\ data, O_1 = O_2)etc....$$

So dividing the problem into the nine subproblems listed above only makes sense if a real probability exists that groups come from the same or adjacent sections of an object. To argue that this classification will be productive we need an earlier assumption that objects contain relatively few convex parts. This means that two nearby sections of edges in an object have a reasonable chance of coming from the same or adjacent convex parts.

As the above equation shows, to calculate these nine probabilities exactly, we would need to know things like the $P(same|rest\ of\ data, O_1 = O_2)$, where *rest of data* does not include type information. This will depend on a variety of factors such as the number of convex sections in each object. We might calculate these probabilities for a specific application, for which we know which objects will appear in scenes. We can not calculate them in general, however, and so this thesis will not discuss their effect.

The Probability of $type_1$ Occurring

We would like to determine the probability of two groups having a $type_1$ relationship given that they come from the same object, and also given all the other variables that describe the groups. We can do this by looking at the probability depending on whether they actually come from the same convex section, adjacent ones, or non-adjacent ones. So we find:

$$P(type_1|rest\ of\ data, O_1 = O_2)$$

$$= P(type_1|rest\ of\ data, O_1 = O_2, same) * P(same|rest\ of\ data, O_1 = O_2)$$

$$+P(type_1|rest\ of\ data, O_1 = O_2, adj) * P(adj|rest\ of\ data, O_1 = O_2)$$

$$+P(type_1|rest\ of\ data, O_1 = O_2, notadj) * P(notadj|rest\ of\ data, O_1 = O_2)$$

$P(type_1|rest\ of\ data, O1 = O2, same) = 1$. We will make some general comments on the remaining parts of the above equation.

Suppose first that we have two groups of edges that come from adjacent convex sections of the same object. This greatly restricts the possible orientations the groups can have. In particular, they cannot have a $type_3$ relationship. This leads us to expect that both $type_1$ and $type_2$ relationships will be more likely.

An example may make this point more clear. Consider two groups, each consisting of a single short edge. The projection of each group consists of half the plane. If we randomly orient the two edges, there will be about a $\frac{1}{4}$ chance that they fall in $type_1$, a $\frac{1}{2}$ that they will fall into $type_2$, and a $\frac{1}{4}$ chance they will fall into $type_3$. If the groups can not fall in $type_3$, but have an otherwise random orientation, they will now have a $\frac{1}{3}$ chance of falling in $type_1$. So, if the only effect of two groups coming from adjacent convex sections were to preclude the possibility that they fall into $type_3$, then we could conclude that this increases the probability that they fall into $type_1$ or $type_2$, in a way we can quantify by studying the probability of randomly oriented groups falling into various types.

The reasoning above, then, makes the simplifying assumption that groups from different convex sections are randomly oriented, except with the constraint that adjacency may rule out certain orientations. Using this assumption, we find that:

$$P(type_1|adj, O_1 = O_2, rest\ of\ data)$$

$$= \frac{P(type_1|O_1 \neq O_2, rest\ of\ data)}{P(type_1|O_1 \neq O_2, rest\ of\ data) + P(type_2|O_1 \neq O_2, rest\ of\ data)}$$

In the section on groups that come from different objects, we will see how to calculate these probabilities. Extending this assumption to groups that come from non-adjacent sections of the same object, we see that it implies that these groups have random orientations with regard to each other, and

$$P(type_1|notadj, O_1 = O_2, rest\ of\ data) = P(type_1|O_1 \neq O_2, rest\ of\ data)$$

This assumption about orientations, however, probably does not accurately describe the world. Consider, for example, groups formed by taking the convex sections of a star. All pairs of such groups fall into $type_1$ of $type_2$, even the non-adjacent ones. More generally, it seems that because two groups from the same object must link up eventually, they are more likely to point in the same direction. This intuition leads to the conclusion that while the above assumption may approximate the right answer, it fails by underestimating the probability that groups of objects will fall into a lower type. It underestimates, for example, the probability that two groups from non-adjacent sections will fall into $type_1$.

The Probability of $type_2$ or $type_3$ Occurring

The analysis of *types* 2 or 3 occurring closely resembles the previous analysis. We want to calculate:

$$P(type_2|rest\ of\ data, O1 = O2)$$

$$= P(type_2|adj, rest\ of\ data, O1 = O2) * P(adj|rest\ of\ data, O1 = O2)$$

$$+P(type_2|notadj, rest\ of\ data, O1 = O2) * P(notadj|rest\ of\ data, O1 = O2)$$

We know that $type_2$ can not occur if the groups come from the same convex section of the object.

We have already discussed the problem of deriving $P(adj|rest\ of\ data, O1 = O2)$ and $P(notadj|rest\ of\ data, O1 = O2)$, explaining why we will avoid it.

If the two groups of edges come from adjacent sections of the object, they must fall into $type_1$ or $type_2$. We have already discussed the probability that they fall into $type_1$, so the probability they fall into $type_2$ is just one minus this probability.

And we have previously discussed the probability that the relationship between two groups falls into $type_2$ given that they come from non-adjacent sections of an object. That is, we will assume the two groups have random orientations, and that:

$$P(type_2|notadj, O_1 = O_2, rest\ of\ data) = P(type_2|O_1 \neq O_2, rest\ of\ data)$$

Similarly, we find that:

$$P(type_3|rest\ of\ data, O_1 = O2)$$

$$= P(type_3|notadj, rest\ of\ data, O1 = O2) * P(notadj|rest\ of\ data, O1 = O2)$$

$$= P(type_3|rest\ of\ data, O1 \neq O2) * P(notadj|rest\ of\ data, O1 = O2)$$

3.4.2 When Groups Come from Different Objects

Suppose we assume that two groups that come from different objects have random orientations. Then calculating the probability of those groups falling into a particular type becomes a straightforward, although not trivial, problem in geometry. As we saw in the discussion of the role of distance in grouping, this idealization of random orientations does not completely hold. However, we will proceed under the assumption that it holds most of the time, and that when it does not hold, its failure does not affect the outcome much.

Recall from our initial analysis of the problem that we want to calculate:

$$P(type_x | rest\ of\ data, O1 \neq O2)$$

We can introduce these problems with a simplified analysis. We will calculate the relevant probabilities when a distance of 0 separates the beginning of each group from its end. This reflects the situation when groups are small relative to the distance that separates them. We will then discuss how to extend these results to the more general case.

$type_1$: the Simplified Problem

This problem becomes conceptually simple. We now have two points, each projecting a cone across the plane. Or the projection may be just a point, if a group has a finite projection. The probability that another small group, randomly located in the image, will fall into the projection of a group just equals the percentage of the image that this projection covers.

Recall that a_1 represented the angle between the two vectors that form the projection of group 1. So, if group 1 is near the center of the image, the chances that the second group falls inside this projection are $\frac{a_1}{2\pi}$. Similarly, the chances of the first group falling in the second's projection are $\frac{a_2}{2\pi}$. Since the two groups have independent orientations, the chances of them having a $type_1$ relationship are $\frac{a_1 * a_2}{4\pi^2}$.

$type_2$ and $type_3$, the Simplified Problem

It turns out that we can express the probability that the projections of two groups do not intersect with the formula:

$$\frac{\frac{3\pi}{2} - a_1 - a_2 + \frac{a_1 a_2}{2\pi}}{2\pi}$$

So the probability that the projections of the two groups do intersect is just 1 minus this value combined with the probability of $type_1$ occurring. Appendix B contains a proof of this.

The Real Problem

We now want to consider the situation in which the group has some size. This thesis will not provide complete solutions to this problem. However, the next sections will discuss how we expect those answers to differ from the simplified ones derived above.

$type_1$, the Real Problem

When groups may also have a length (expressed by the variables l_1 and l_2), this affects the probability of one group falling in another's projection in a number of ways. In the

simplified problem, if group 1 fell into group 2's projection, this did not affect the probability of group 2 falling into group 1's projection. We will not be able to count on this in the real problem. In fact, every orientation of group 1 will not only determine if group 2 falls in its projection, but will also determine the likelihood that it will fall in group 1's projection.

This section will make the point that the greater the projections of the two groups, the greater the chances that they will have a $type_1$ relationship. On the other hand, the larger the groups, the less the chances that they will have this relationship. Furthermore, this section will suggest some bounds on the probability of $type_1$ occurring, in the case where both groups have infinite projections.

Both the starting and ending point of group 1 must fall in group 2's projection for them to have a $type_1$ relationship. Instead of overlapping just one point, group 2's projection must now sweep over a whole slice of the plane. We will look at the impact of this effect when group 2 still has a length of 0, and then when it does not.

If group 2 began and ended at the same point, then the lines from this point to where group 1 began and ended would form an angle, call it α. α would have a maximum value when group 1 faced group 2 directly. At that point, α would equal $2\arcsin\frac{l_1}{2d}$. As group 1 tilted with respect to group 2, α would diminish to 0. The probability of group1 falling in group 2's projection would be $\frac{a_2-\alpha}{2\pi}$, or 0 if $\alpha > a_2$.

When group 2 also has a length, however, we must consider the fact that at distance d from group 2, its projection sweeps out a larger area. For greater accuracy, we should consider not the angle of a group's projection but its total size. We ignore this subtlety, however.

We must also contend, however, with the complexity that the chances of group 1 falling in group 2's projection depends not just on group 2's orientation, but also on group 1's. This means that the probability of both groups falling inside each other's projections is not simply the product of two simple probabilities. If group 1 falls in group 2's projection, where in the projection it falls will determine how oblique an aspect group 2 presents to it. Furthermore, we must also now consider a_3 and a_4, the angle between the projection of a group and a line joining its beginning and end. For some values of a_3, for example, the fact that group 2 falls in group 1's projection may imply that group 1 presents only a narrow view of itself to group 2. These dependencies make a precise calculation difficult.

However, we can calculate bounds on this probability. Based on the above analysis, we see that the best chance group 1 has of falling in group 2's projection occurs when it appears to group 2 head on, as a point. The worst chance occurs when it faces group 2 directly, presenting an aspect of length l_1. This suggests that the probability of the groups having a $type_1$ relationship has the simplified analysis of the last section as an upper bound, and has as a lower bound the product of the two minimum probabilities treated independently.

This would tell us that:

$$P(type = type_1 | rest\ of\ data) < \frac{a_1 * a_2}{4\pi^2}$$

and

$$P(type = type_1 | rest\ of\ data) > \frac{a_1 - (2 \arcsin \frac{l_2}{2d})}{2\pi} * \frac{a_2 - (2 \arcsin \frac{l_1}{2d})}{2\pi}$$

$type_2$ and $type_3$, the Real Problem

We do not analyze this more complicated problem. We just point out that larger groups will have larger projections, with an increased chance of intersecting.

3.5 The Distance to Intersections

This section will argue, somewhat superficially, that when two groups have a $type_2$ relationship, the greater the distances to the place where their projections intersect, the less the chances that the groups come from adjacent sections of the same object. These distances, denoted by int_1 and int_2, indicate how long the two convex sections that produced these groups must be in order to intersect. The longer these sections must be, the larger the scale of the object that produced them. This essentially restricts the possible ways that a single object could have produced the two groups.

int_1 denotes the minimum distance from the beginning or end of group 1 to the projection of group 2. The projection of group 2 tells us the possible location of any obscured part of the convex section that produced group 2. So, if in the scene the two groups come from adjacent convex sections of the same object, then some obscured portions of group 1 and group 2 must continue the groups at least until the point where their projections intersect.

This gives us two reasons for expecting low values of int_1 and int_2 from groups that really come from adjacent sections of the same object. First of all, we expect the convex sections of the object to have shorter, rather than longer lengths of perimeter missing from the image, because we expect in general that shorter, rather than longer sections of perimeter will fail to appear in the image. Secondly, suppose int_1 has a value of 10 units. Then group 1 could come from any section of any object scaled so that it has a diameter greater than 10 units. But if int_1 has a value of 20 units, then only sections scaled to exceed 20 units in diameter could have produced group 1, if group 1 does indeed come from a convex section adjacent to the one that produced group 2. By limiting the number of possible scenes that might have produced an image with adjacent sections causing groups 1 and 2 we reduce the chances that adjacent sections did produce groups 1 and 2. Both of

these reasons support the hypothesis that the chances of two groups coming from the same object decrease as int_1 and int_2 increase.

On the other hand, a complete analysis would require us to examine the distributions of int_1 and int_2 when the groups come from different objects, as well as more thoroughly analyzing these distributions when the groups come from the same object. We do not attempt such an analysis here.

3.6 Future Work

In paring the grouping problem down to just two factors, distance and orientation, we made many simplifications and threw away a lot of information present in images. The fact that we can use this theory of grouping to build a successful recognition system demonstrates the power of these two constraints. However, we should expect to perform better grouping by analyzing a more realistic model of the world, and by taking into account more information. These two kinds of additions work together, when we try to make use of more information, we need a richer model that tells us how to use that information. We can see this if we try to enhance grouping so that it can take advantage of three dimensional information, texture or color information, or shape information.

We would like a grouping system that works well in three dimensional domains, not just on two dimensional scenes. Chapter 9 will show some examples that indicate that GROPER's grouping system does well on images created by three-dimensional objects. But surely we can do better with a theory of grouping that takes into account the differences between two and three dimensional scenes, or uses depth information derived from images.

Three dimensional scenes produce two dimensional images with different characteristics than the ones produced by two-dimensional scenes. For example, nearby things tend to appear in front of and larger than far away things. This might, for example, result in even shorter sections of occluded perimeter than with the situation analyzed in this chapter, because nothing occludes the nearest and largest objects. A more thorough analysis of the situation would tell us how it differed from the one discussed in this chapter.

We also might want to make use of three dimensional information determined from the scene. For example, we might use stereo or motion clues to determine the distance from the viewer of different sections of the image. This could provide an important source of clues for grouping, but to use this information we would need to know something about how three dimensional scenes behave. For example, suppose we see a nearby patch on our left and a more distant patch on our right. We might expect, due to this distance, that it is less likely that these two patches come from the same object than if they were equally distant. But how much less likely? And what other factors influence this likelihood? Suppose the

patch on our left slopes away from us at a rate that lines it up exactly with the location of the patch on the right. A theory of three dimensional scenes would tell us how to make use of this type of depth information to perform grouping.

We would also like to take advantage of color, shading and texture information when we perform grouping. This type of information appears to provide important clues in deciding whether sections of an image come from the same object. For example, we would expect two sections of an image with the same texture to have a greater likelihood of coming from a single object than if they had different textures. But to make use of this information, we need a much more elaborate model of objects and of the image formation process.

We probably can draw the conclusion that similar textures more often come from the same object than do dissimilar ones, because we assume that if one section of an object has a certain texture, or color, then that texture or color is much more likely to show up in other parts of the object. But how much more likely? Does the distance between the parts of the object effect this likelihood, and if so, how? What about small changes in texture or color? Is a pink section of the image likely to come from the same object as a red section? and how does this likelihood differ when both sections are red? To answer these kinds of questions we need to have some kind of model of how objects are colored and textured, and how colors and texture change or repeat over the surface of objects.

We also need to know how lighting effects can influence this process. For example, if two sections of an image appear to have different textures, we need to know if this means that the sections of object that produced them have different textures, or whether a difference in lighting might have produced this effect. Land [17] and others have proposed theories of computation to explain how we may answer this type of question for color and shading. We would like to approach the problem of texture in the same way, with a theory that tells us how to determine the chances that two image sections come from pieces of material that have similar colors and textures.

Shape information can also provide important clues in determining the likelihood of groups coming from the same object. For example, if two groups have symmetric edges, this might incline us to think that they come from the same object. And, in the simplest case of symmetry, we would like to know how the presence of parallel edges will influence this likelihood. Lowe[18] has analyzed this problem for a world model containing randomly oriented, three-dimensional objects. Also, perhaps asymmetric but still similar shapes will provide important clues. Judging the presence of such relationships does not cause much of a difficulty, but knowing how to make use of them does. To do so, we would like both a more realistic model of the shape of objects, and of their orientations.

The model of objects used in this chapter assumes essentially random objects with random orientations. In such a world, symmetry and similarity of shape occur only through

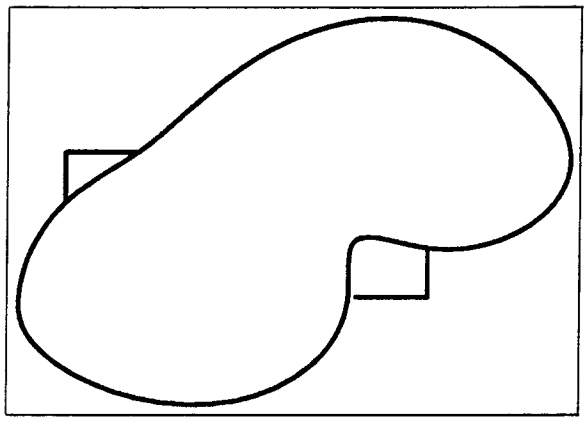

Figure 3.13: The four straight edges group better together because the other edges account for the missing perimeter of an object that produced those edges.

chance, and so occur only rarely both between groups that come from different objects and from the same objects. In the real world, however, symmetry and similarity of shape occur all the time. On the one hand, a more realistic model of objects will tell us that they often contain symmetric sections, having parallel edges especially often. Also, often the same shape occurs in different parts of an object. People, for example, usually have symmetric faces, and two similar hands that are not oriented to appear symmetric. On the other hand, the orientation of objects often conspires to produce symmetry and similarity between different objects. Consider parallel edges. My office now contains scores of horizontal and vertical edges. Some pairs of these come from a single object, but most come from a wide variety of different objects, because scenes constructed by humans tend to contain objects oriented to produce many horizontal and vertical edges. We might not want to conclude that the parallelism of two edges makes them likely to come from a single object. Also, a scene may contain many similar shapes that come from different objects. A forest contains many leaves for example. Many problems remain in constructing an accurate theory of how to make use of this kind of shape information, because to do so we must understand how to model the regularities in our world.

Finally, this chapter only considered the relationship between two groups of edges in determining the probability that they go together. But other edges in the scene will also effect this probability. For example, if other edges surround the area in between the two groups, this might make it look more like the two groups come from the same object (see figure 3.13). This is because the other edges outline an object that would cover up the area between the two groups, providing an explanation for the distance that separates them. On

Figure 3.14: These two groups go well together.

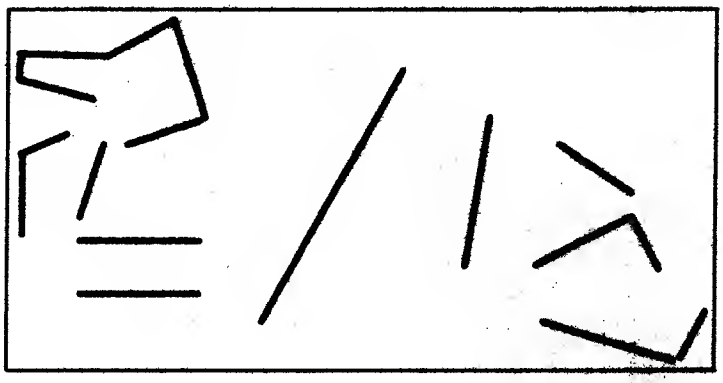

Figure 3.15: But when considered as part of a scene, the groups from the previous illustration may not go as well together.

the other hand, if other edges in a scene go particularly well with one group of edges this might make it less likely that that group belongs with a different group of edges. We need to check not just whether a pair of groups go well together, but whether this pairing will lead to a good global interpretation that explains all the edges in the image. Figure 3.14, for example, contains two groups of edges that go well together when considered in isolation. But when considered as part of an image (figure 3.15), we see that combining these edges will not lead to a satisfying interpretation of all the edges in the image. Moving from a local consideration of pairs of groups to a global consideration of many groups creates challenging new problems.

3.7 Conclusions

This chapter has focused on determining how to use distance and orientation information when performing grouping. It has reached a fairly simple conclusion about the effect of distance: the greater the distance between two groups of edges, the less likely they are to come from a single object. Furthermore, it has provided one of the reasons this might be true, and made conjectures about bounds on how much this likelihood changes with distance. The chapter has also told us how to deal with short distances separating two groups of edges; this probability depends on the general likelihood of occlusions occurring in a scene.

The analysis of orientations does not lead to quite as straightforward a solution. We have profitably divided orientations into three classes. Other factors being equal, we find that some *types* indicate that groups come from the same object more than do others. Furthermore, we see the general effect that changes in the angles and sizes of the groups have on the likelihood that the groups come from the same object. In addition, we have some bounds on parts of this probability. We have found that a full analysis of these probabilities would require us to make assumptions about the kind of objects contained in the library, as well as making assumptions about the kind of scenes we will encounter. However, we have a sound basis for heuristically choosing probability distributions that have the correct general shape.

As the chapter on the grouping algorithm will make clear, we use the results of this chapter to make some informed guesses at the probabilities we want. Although not optimal, the heuristics chosen do result in a successful grouping system. Without this analysis, the choice of heuristics would not be obvious, at least it was not obvious to the author. So the accomplishment of this chapter is that it provides inspiration for a good choice of grouping heuristics, and a justification for arguing that these heuristics reflect the effect of real constraints of the world.

Chapter 4

The Psychophysics of Grouping

4.1 Introduction

The theory of grouping presented in chapter 3 predicts that some combinations of edges have a greater likelihood of coming from the same object than do others. When people look at images containing edges, they see certain combinations of edges grouped together. This chapter points out that the theory of grouping derived in chapter 3 agrees with human grouping phenomena. This leads to the hypothesis that people perform grouping as a first step towards recognizing objects. According to this hypothesis, they group together the edges and other features in an image that have a particularly good chance of coming from a single object. Furthermore, we expect that evolution has equipped people to perform this task quite well. Following this hypothesis, we would expect any correct theory of grouping to accurately predict human behavior. Correct predictions based on the theory of chapter 3 therefore lend support to that theory, telling us that even though the theory was derived from a simplified world, it may still accurately reflect the real world. Furthermore, the theory of grouping provides a deeper explanation of human grouping phenomena. We had hypothesized that people group together edges likely to come from the same object. But now, at least in some cases, we know *why* these edges are more likely to come from one object.

This chapter makes use of only informal psychophysical evidence. It presents some images, and claims that "obviously" people will tend to group together one pair of groups of edges over a different pair. "Group together" means that the viewer will experience a sense that certain combinations of edges go together well, or that if asked, the viewer would affirm that those combinations of edges seem more likely to come from the same object than another combination of edges. The reader, of course, will judge for him- or herself whether these claims are obviously true.

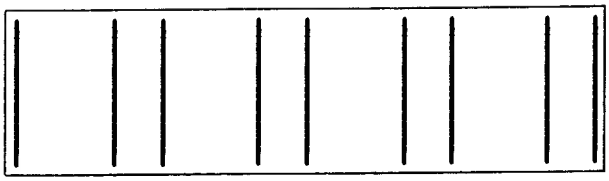

Figure 4.1: People tend to form groups out of the nearby edges rather than the more distant ones.

The examples given in this chapter try to isolate the effects of distance and relative orientation from other effects. For that reason, we try to avoid using images in which other factors might play an important part. For example, we know that people tend to group together parallel, symmetric, or co-linear edges. When comparing two different pairs of groups of edges, we make sure that either no pair contains such a quality, or that all pairs contain them equally.

In the past, psychologists, and more recently, computer scientists, have investigated grouping phenomena. Lowe[18] offers an excellent summary of that research, which this paragraph draws upon. The Gestalt psychologists originally investigated grouping phenomena. Wertheimer[26] noticed that people will group together parts of an image, and suggested five different principles to explain the phenomena he observed. For example, he suggested that people group together nearby things, and similar things. The gestaltists also suggested that people group together those parts of an image likely to come from a single object. From a computational perspective, Witkin and Tenenbaum[27] argue that people's perceptions make salient the aspects of an image least likely to occur accidentally. Parts of an image that have a relationship unlikely to happen by chance probably have a related underlying cause. Lowe[18] also follows this line of reasoning. He argues that nearby, parallel edges have a high probability of coming from the same object because such edges occur only rarely by chance when the edges come from different objects, but occur more often when the edges come from the same object. The type of reasoning utilized in chapter 3 obviously draws much inspiration from these approaches. This chapter will make use of the fact, noticed by Wertheimer, that people group together nearby things. It will also discuss the relationship between the orientation constraint developed in chapter 3 and other previously studied grouping phenomena.

4.2 Proximity

All other factors being equal, people tend to group together nearby edges. Figure 4.1

shows a series of parallel edges. People see the nearby edges forming pairs. One might instead see the adjacent, but more distant edges forming pairs. These alternate pairs would differ only from the pairs people do see in the distance separating the edges. That people do not see the image in that way shows that people group together the closest edges in the image.

This agrees with the theory of grouping already presented. That theory suggested that nearby edges in an image have a greater likelihood of coming from the same object than do distant edges. Of the two rival ways of interpreting the image in figure 4.1, the way that people see the image requires an explanation for a series of short sections of perimeter that did not produce edges. The alternative grouping of edges requires an explanation for longer sections of missing perimeter. The theory of grouping explains why people should prefer the first possibility.

As mentioned before, Wertheimer noticed this grouping phenomena. He did not argue explicitly that it arises because nearby edges often come from the same object, however. Lowe does present a rationale for the use of grouping based on the proximity of edges. He takes quite a different approach to the one presented in this thesis. He argues that by grouping together nearby edges that have other qualities, such as parallelism, we can reduce the complexity of recognition by limiting the number of groups we must consider. He does not argue that some edges are more likely to come from a single object simply because they are nearby.

4.3 Orientation

The discussion of orientation in the theory of grouping allows us to make a number of predictions about which collections of edges will group best together. Although the theory does not tell us how to compute the exact probability that a pair of groups comes from the same object, it does offer us some ways to compare different pairs of groups. For pairs of groups with the same type of orientation, we know how altering certain variables will alter the probability that the pairs come from the same object. For example, we know that the larger the angle of projection of two groups, the less unlikely it is that a random orientation would produce a $type_1$ relationship. So, all other factors being equal, two groups with a larger angle of projection and a $type_1$ relationship will be less likely to come from the same object then two similar groups with a smaller angle of projection. We can also show that groups with a $type_1$ orientation have a greater chance of coming from the same object than do groups with a $type_2$ relationship, and that groups with a $type_3$ orientation have the least chance of all. Having made these predictions we can then see if they agree with the way people group edges. They do.

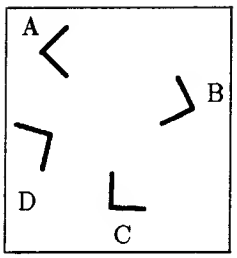

Figure 4.2: Groups A and B have a $type_1$ relationship, and seem to go better together than groups A and C, which are the same, but oriented to have a $type_2$ relationship. These in turn go better together than groups A and D, which have a $type_3$ orientation.

The theory of grouping implies that, all other factors being equal, a pair of groups with a lower type number describing their orientation will have a greater likelihood of coming from the same object. This is because, for example, a $type_1$ relationship always occurs between groups from the same convex section, and can also occur when the groups do not come from the same convex section of an object, while a $type_2$ relationship can only occur when the groups come from different convex sections of an object. In other words, lower types have more ways of occurring when groups come from the same object. Appendix C proves this.

Figure 4.2 shows a group of edges, labeled A, paired with three other groups. The pairs have different types of orientations, but otherwise they are the same. As predicted, the groups in $type_1$ seem to go best together, the groups in $type_2$ go next best together, and the groups with a $type_3$ orientation seem least likely to come from a single object.

The theory of grouping also makes some predictions that allow us to compare different pairs of groups with the same type of orientation. In particular, we will look at how changing the angle of the groups' projections will effect the likelihood that they come from the same object. And when two groups have a $type_2$ relationship we can predict how a change in the distance to the area where their projections intersect will affect this likelihood.

If two groups have a $type_1$ orientation relationship, reducing their angle of projection should increase the probability that they come from the same object, if all other factors remain equal. If a group has a smaller angle of projection, then the other group has a smaller chance of falling inside that projection, if randomly oriented. So a smaller angle of projection decreases the probability of a $type_1$ orientation occurring if the groups come from different objects. It will also decrease the chances of this orientation occurring if the groups come from adjacent or non-adjacent sections of the same object. But, because it does not reduce the chances of this orientation occurring if the groups come from the same convex

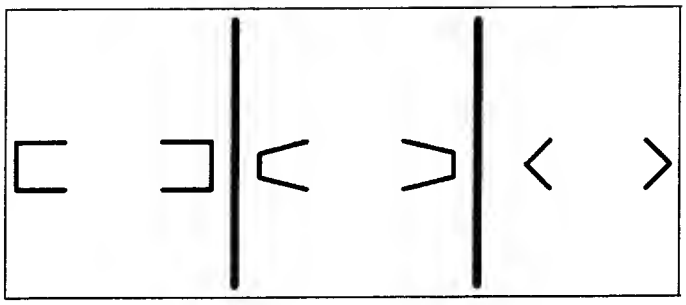

Figure 4.3: Three pairs of groups with a $type_1$ orientation. The pairs have the same relationship, but different angles of projection.

section of an object, overall $\frac{P(type_1|O_1=O_2)}{P(type_1|O_1\neq O_2)}$ increases. Figure 4.3 provides some examples to demonstrate that groups with $type_1$ relationships seem to go better together when they have smaller angles of projection.

This analysis explains certain types of very strong grouping phenomena in humans. Suppose we have an image consisting of the two ends of a rectangle, separated by some distance, as in the left-most pair in figure 4.3. These two groups of edges seem to have an extremely strong bond. We can explain this bond as the limiting case of the situation considered above. Those two groups have a zero probability of having a $type_1$ orientation if randomly located, because this orientation requires their ends to line up perfectly. The only plausible explanation for this orientation, according to the theory of grouping, is that the two groups come from the same convex section of the same object.

The theory of grouping has also predicted that when two groups have a $type_2$ relationship, increasing the distance to the intersection of their projections will decrease the probability that they come from the same object. Figure 4.4 provides examples of pairs of groups that differ only in the distance to the intersection of their projections. These images appear to confirm the predictions of the theory of grouping.

Past work has not paid much attention to grouping that results from the orientation of edges. This may be due to the fact that such grouping phenomena does not produce nearly as striking an effect as do other phenomena. Figure 4.5 contains examples of the grouping phenomena discussed by Wertheimer. These examples all cause strong grouping effects. In contrast, a pair of groups with a $type_2$ relationship and short distances to the intersection of their projections do not go together that strongly, just more strongly than they would with long distances to their intersection. For this reason, most work on grouping has focused on special relationships, such as symmetry, which cause particularly strong grouping. While these effects are important, they do not allow us to evaluate arbitrary collections of edges

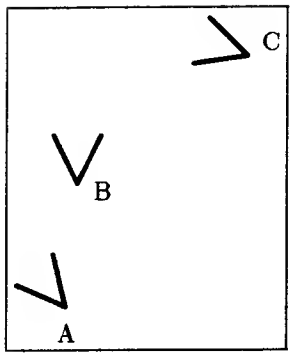

Figure 4.4: Groups B and C seem to go together better than groups A and B. The same distance separates each pair, but the distance to the intersection of their projections differs greatly.

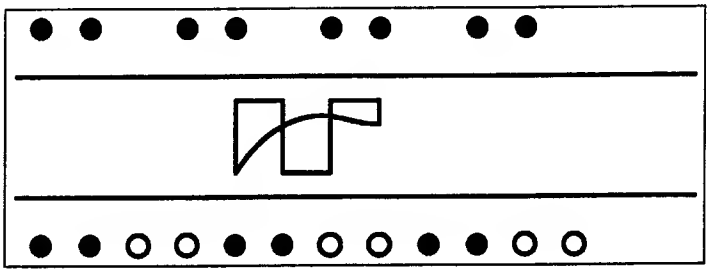

Figure 4.5: Three of the striking grouping phenomena discussed by Wertheimer. At the top, we group together the nearby dots. In the middle, we consider the curved edge as a single contour. We do not see three separate closed areas with three separate curved ends. On the bottom, we group together the similar dots.

to see how well they go together. It is because this thesis aims to find large groups of edges in any kind of image that it has needed to examine less forceful factors in grouping.

4.4 Future Work

This chapter has presented some psychophysical predictions we can derive from the theory of grouping, and shown that they match experience. This attempts to show that an analytical examination of grouping can help to explain human grouping phenomena. But we can also take a different approach. If we accept the hypothesis that human grouping mainly results from an effort to find sections of an image that all come from the same object, then we can use human grouping as a basis for determining how to perform grouping with a computer. By carefully analyzing the way people group edges, we can resolve some issues left fuzzy by the theory of grouping.

For example, experiments that simultaneously vary the distance between two groups and their relative orientation could tell us exactly how to combine these two factors to determine the probability that two groups come from the same object. In chapter 3 we only derive bounds on the way this probability changes with distance, or with the angle of projection of groups. Based on these bounds, we can not always accurately choose one pair of groups over another. If one pair is closer together, but has a $type_1$ relationship with larger angles of projection, experiments could tell us how people weight these two factors to choose one pair over another. We might also use psychophysical experiments to determining exactly how changes in the distance to the intersection of projections should change the probability distributions we want. In general the theory of grouping provides the general shapes of distributions, but experiments might provide more complete and accurate results.

Psychophysical experiments might also test the fundamental hypothesis that people use grouping as a first step in object recognition. We might present people with images in which the edges they naturally group together do not really come from the same object, and see if this slows down object recognition.

Chapter 5

A Grouping Algorithm

5.1 Introduction

Once we have a theory of grouping, we still need to determine how to apply that theory to create a grouping algorithm. The theory of grouping presented previously, while providing some clues about how to compare different pairs of groups, does not tell us everything we need to know. This chapter describes how GROPER makes such a comparison based on these clues. But such a test does not tell GROPER how to build up groups of edges large enough to help it recognize an object. So this chapter also describes how to use the theory of grouping to develop some short-cuts for creating helpful groups.

GROPER performs grouping in three stages. First, it combines nearby edges into convex sections. These convex sections of edges then become the new primitives which GROPER uses for grouping; it never looks back at the initial edges. GROPER forms these convex sections because they fit the criteria for good groups which we can deduce from the theory of grouping. Secondly, GROPER considers every pair of these primitive groups to find the two groups which seems to go together best, and then combines them to form a new group. Performing this step involves deriving a comparison metric from the grouping theory. Thirdly, GROPER sometimes wants to extend one of these new groups, finding a third group which goes well with it. This may happen when indexing into the library of models reveals that a pair of groups match a number of different known objects. Together, these three stages feed the recognition section of GROPER, and respond to its results.

5.2 Forming Convex Sections

GROPER begins the grouping process by quickly forming connected, or nearly connected, convex sections of edges. This speeds things up a lot. It is a computationally inexpensive

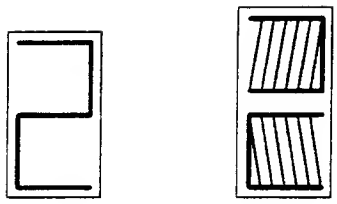

Figure 5.1: On the left are five image edges. On the right, the two groups formed by GROPER out of these image edges, displaced slightly from their location on the left. The shaded area indicates the sides of the edges on which GROPER has decided the figure lies. Notice that an edge can appear in more than one group.

process that allows GROPER to then consider combining groups of two, three or more edges, instead of dealing with only single edges. Combining a pair of single edges results in two edges that will usually match too many different objects to be helpful. But combining a pair of convex sections may result in something quite useful for recognition, because it contains more information. Also, this process creates groups of edges about which we can make figure/background judgments. With a single edge, we cannot determine this. But two connected edges appear convex only when we assume that the object lies on a particular side of the edges. However, beginning grouping this way also has a drawback. We must make sure that these convex sections actually contain edges likely to have come from a single object. GROPER uses these convex sections as its new primitives. Once it forms these sections, it never looks at the isolated edges again. So if GROPER incorrectly groups together edges that do not belong together, it may never be able to use those edges to help it find objects.

As input, GROPER uses straight line segments which approximate the edges that an edge detector has found in the image. This thesis will not discuss either the edge detection or the straight line approximation steps; many standard methods exist for solving both problems and GROPER makes no contribution in either area. For each line segment GROPER does not know on which side the object lies, and which side forms the background of that object. As output, GROPER produces groups of line segments. Each group designates an inside and an outside for each edge in the group. Each group of edges forms a connected, or nearly connected, convex curve. One edge may appear in more than one group, having a different inside and outside in different groups. Figure 5.1 shows some line segments and the two convex groups formed from them.

GROPER performs this step in two phases. First, it forms nearly connected "strings" of line segments. Then, it breaks up these strings into convex sections. GROPER forms

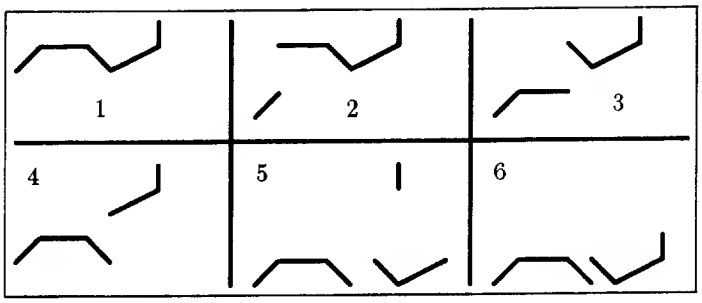

Figure 5.2: This shows the decomposition of a string of connected edges into two groups. Notice that in part five a new convex group is started using the same edge which ends the last convex group.

strings of line segments by connecting two segments whenever they have end points closer to each other than to any other end points of line segments. By any other end points, we also include the other end points of the two line segments considered for connection. So GROPER never connects two line segments if the distance between their end points exceeds the length of either line segment. GROPER "connects" two line segments by just putting them together in a list, ordered appropriately. Each segment appears in exactly one list (although GROPER may later break this list so that it appears in two groups), and a list may contain one line segment, or more than one. A line segment may connect to two others, one at each end. We do not allow an end point to connect to more than one other segment to avoid creating too many strings, but, of course, sometimes this leads to mistakes.

GROPER then determines the figure and background sides of each edge by breaking the strings of line segments up into convex groups. GROPER simply follows the edges along a string. As long as the edges form a convex group, GROPER does not touch them. Adding an additional edge might make a convex interpretation impossible. For example, three edges might make a Z shape. In that case, GROPER takes the initial group of edges, and makes them a separate group. The first two edges of the Z would form a separate group, for example. GROPER then takes the last of these edges, adds the new edge on, and forms a new group containing two edges so far. The last two edges in a Z would also form a group. This results in one edge belonging to two groups, having a different figure/ground interpretation in each group. This second group of two edges may then have more edges added to it as GROPER continues to explore the rest of the edges in the string. Diagram 5.2 gives an example of this process.

This initial step produces groups of edges which have a particularly good chance of

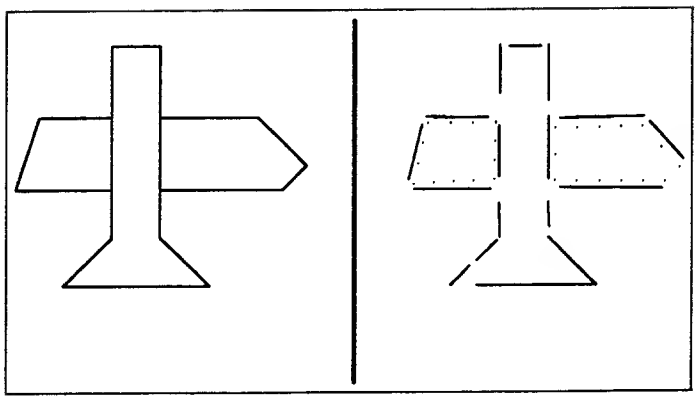

Figure 5.3: On the left are the perimeters of two different objects, with a section missing where one lies on top of the other. On the right, the type of edges which GROPER's pre-processing typically produces. The dotted lines indicate two convex, nearly connected groups of edges.

coming from the same object. The previous chapter told us that the less the distance which separates two edges in an image, the greater the chances that they come from a single object. By connecting together the closest end points, we choose the most likely connections. Furthermore, we take into account the null hypothesis that a line segment represents the only section of an object's perimeter which produced an edge in the image. In that case, the length of missing perimeter equals the length of the line segment. So we only connect line segments whose distance apart is less than that distance.

Also, the previous chapter makes the convex interpretation of adjoining line segments seem the most likely. We assume that objects have relatively few points of concavity. This makes it more likely that two nearby edges come from the same convex section of an object than from adjacent convex sections. If we interpret the edges so that they form a concave shape, then they would have come from two adjacent convex sections of an object.

We might put forward an additional argument in favor of this grouping step, as well. Some authors (see Schwartz and Sharir[22], for example) have used the fact that in some circumstances, convex sections of edges are extremely likely to come from a single object. Two objects together can produce a convex outline only when they fortuitously line up. Two squares, for example, form a convex contour together only when they each have a corner located at the same point. We do not make use of this argument, however, because it holds true only in the presence of conditions which this thesis does not assume. If we do not know figure from ground, or if the perimeters of objects do not produce continuous edges, then two different objects may often combine to produce connected, or nearly connected

convex contours. For example, two occluding objects always produce a pair of convex edges, if we mistake figure and background in the edges from both objects. If some of an object's perimeter fails to produce edges, this can happen in other ways as well. Figure 5.3 shows the complete perimeter of two occluded objects and the edge line segments which GROPER's pre-processing steps typically produce from such a scene. The dotted lines show two collections of convex, nearly connected edges. Neither of them comes from a single object.

Making these convex groups of edges greatly reduces the amount of work left to the grouping module. Without these groups we would have to look deeply into a search tree of possible combinations of edges. Furthermore, each edge would have two possible interpretations, since we would have no way to tell the figure side of a single edge from its background side. Instead we usually only need to find the best combination of two of these convex groups to find an object. Chapter 9 provides data which shows that groups of four or five edges usually match few objects in the libraries GROPER uses. And, even though a line segment can appear in two groups, GROPER usually produces fewer primitive groups of edges than line segments, since many groups contain three or even four edges. So this initial simple grouping step greatly reduces the amount of work needed to find objects by quickly combining the edges most likely to come from the same object.

This section of GROPER puts together the best small groups it can. It does make mistakes. But this phase of GROPER works quickly. For an image with about one hundred edges, this phase of GROPER takes less than ten seconds to perform on a Symbolics 3600 Lisp Machine. Furthermore GROPER performs this task with an extremely inefficient algorithm which carries out an $O(N)$ process in $O(N^2)$ time. We have not bothered to make this phase of grouping faster, since it still represents only a small fraction of the total recognition time.

5.3 Combining Convex Sections

Once GROPER has formed these primitive convex groups, it then looks for larger groups which can help it to recognize an object. A single convex group may prove large enough to allow recognition of an object. But usually, GROPER must combine at least two of these groups. GROPER does this in a quite simple way. It considers every possible pair of groups of edges, to find the pair most likely to come from a single object. The difficulty of this task arises in determining how to evaluate these pairs of groups. To do this, GROPER calculates an evaluation metric based on the theory of grouping discussed in the previous chapter.

First of all, however, GROPER examines the simple convex groups to see if any of

them might allow it to recognize an object. GROPER can use a simple group to index into its library of objects, and perform verification if the results look promising. To produce promising results, a group must match only a few different sets of edges among the known objects. If a group of edges matches many sets of object edges, GROPER goes back to its grouping module to try to extend the group. Since any collection of three or fewer image edges will usually match quite a few objects, GROPER does not even bother to perform indexing with such small groups. GROPER does index into its library of objects using any simple groups with four or more edges, and can recognize objects in this way.

After this process, GROPER next looks for the most promising pair of simple groups. To find this pair, GROPER looks at the distance and relative orientation of each pair of simple groups. For each pair GROPER calculates four probabilities: $P(d|O_1 = O_2), P(d|O_1 \neq O_2), P(type, int1, int2|O_1 = O_2, d, l_1, l_2, a_1, a_2, a_3, a_4)$, and $P(type, int1, int2\ |O_1 \neq O_2, d, l_1, l_2, a_1, a_2, a_3, a_4)$, and combines them to determine a metric which reflects the probability that that pair of groups came from the same object.

5.3.1 Distance, the Same Object

First of all, GROPER estimates $P(d|O_1 = O_2)$. It does this with the function:

$$P(d|O_1 = O_2) = \frac{(MaxDiameter - d)^2}{\frac{MaxDiameter^3}{3}}$$

where MaxDiameter is just the size of the image. This function reflects the occlusions which arise in a library of objects which each have a uniform distribution of distances between pairs of points, when we then take those objects at uniformly distributed scales. That is, we take a uniform distribution, average it over all scales, then square it. Figure 5.4 compares this distribution to the worst case distribution produced by a circle, and to the distributions created by a few randomly chosen polygons.

5.3.2 Distance, Different Objects

The theory of grouping has also told us that we can find the distribution of distances between groups from different objects by using a linear combination of the above distribution, and a distribution which reflects the distances between randomly oriented groups of edges. For small groups of randomly oriented edges, we can approximate the distribution of the distances between them with the function:

$$P(d) = \frac{2\pi d}{\pi MaxDiameter^2} = \frac{2d}{MaxDiameter^2}$$

for $d \leq MaxDiameter$. And the experiments depicted in figure 3.12 show that this distribution does not differ much from the distribution when the groups are not small. So, if we

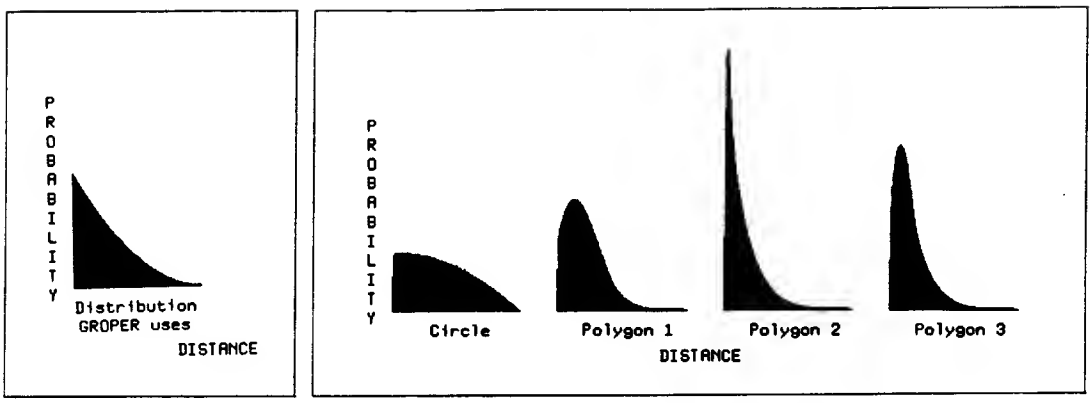

Figure 5.4: On the left, the probability distribution which GROPER uses for the expected distance between groups of edges from the same object. Then, the distribution we would expect from a library of just circles, or just one of the three random polygons shown in figure 3.7.

linearly combine this distribution with the one given in the previous subsection, we find:

$$P(d|O_1 \neq O_2) = k * \frac{(MaxDiameter - d)^2}{\frac{MaxDiameter^3}{3}} + (1-k) * \frac{2d}{MaxDiameter^2}$$

That is, we approximate the distribution of distances between groups from different objects by combining two distributions. The first reflects what happens when the groups were created by the intersection of the two objects. In this case, we have the same distribution as when the groups come from the same object. The second distribution increases linearly, and reflects what happens when the two groups have independent, random locations. The way in which we combine these two distributions should then reflect the likelihood of occlusions occurring in the scenes GROPER will examine. GROPER uses the arbitrarily selected value of .2 for k, which seems to work well.

5.3.3 Orientation, the Same Object

GROPER calculates the probabilities of different orientations occurring based on their For example, if two groups have a type one orientation, recall that:

$$P(type_1|O_1 = O_2, d, restofdata) =$$

$$P(type_1|same, restofdata) * P(same) + P(type_1|adj, restofdata) * P(adj)$$

$$+P(type_1|notadj, restofdata) * P(notadj)$$

where "same" denotes that the two groups originally came from the same convex section of an object, and "adj" and "notadj" have comparable meanings. So to calculate all the

probabilities we need, we must know the chances that two groups of edges, randomly selected from one object, come from the same, adjacent, or non-adjacent groups of edges, and we must know the nine conditional probabilities which combine these three ways the groups can originate with the three orientation types which reflect the way they look.

We arbitrarily choose the value $\frac{1}{3}$ for $P(same), P(adj)$ and $P(notadj)$. Although this probably does not reflect reality very well, this does not matter too much. We want probability distributions with the right general shape, but we do not care if some of their details are not optimal. A choice of $\frac{1}{3}$ guarantees that GROPER takes each of the three possibilities into account in determining the distribution it wants.

Of the remaining nine probabilities, four have obvious values. For example, $P(type_1|$ $same, restofdata) = 1$. That is, groups of edges which come from the same convex section cannot look as if they could not have come from the same convex section. Of course, in determining the type of two groups, we must take possible error into account. If two groups really do not fall inside each other's projections, but allowable amounts of error would have them falling inside each other's projections, GROPER considers them to have a $type_1$ relationship. So if they do have a $type_1$ relationship, the groups really could not have come from different convex sections of the same object. Similarly, we find that $P(type_2|same) = 0$, $P(type_3|same) = 0$, and $P(type_3|adj) = 0$. We must approximate the remaining five probabilities. Chapter 3 concluded that:

$$P(type_1|adj, O_1 = O_2, restofdata) = \frac{P(type_1|O_1 \neq O_2)}{P(type_1|O_1 \neq O_2) + P(type_2|O_1 \neq O_2)}$$

$$P(type_1|notadj, O_1 = O_2, restofdata) = P(type_1|O_1 \neq O_2)$$

and

$$P(type_2|notadj, O_1 = O_2, restofdata) = P(type_2|O_1 \neq O_2, restofdata)$$

And, of course:

$$P(type_2|adj) = 1 - P(type_1|adj)$$

and

$$P(type_3|notadj) = 1 - P(type_2|notadj) - P(type_1|notadj)$$

In addition to the type of orientation, GROPER also makes use of the distance to the intersection of projections when two groups have a $type_2$ relationship. Recall that we call these distances int_1 and int_2. This might seem like an esoteric piece of information of which to make use, particularly since we do not have a well developed theory of int_1 and int_2. However, the informal psychophysics discussed in chapter 4 should convince the reader of the importance of these variables. And using this information significantly improved GROPER's grouping.

When using intersection information, GROPER first looks for some special cases which it treats separately. First of all, GROPER sets a maximum value on the size that an object part can have. It sets this value at about half of the size of the image. If either int_1 or int_2 exceeds this size, then the two groups can not come from adjacent sections of the same object after all. Secondly, if only a short distance separates the two groups, GROPER ignores the values of int_1 and int_2 as unenlightening. For example, if two groups touch where they end, their intersection distances will always have the value zero, which tells us nothing about the likelihood that they came from the same object. Even when the groups just end near each other, almost any orientation of the groups will produce int_1 and int_2 with values close to zero. So the random orientations which the groups presumably have when they come from different objects will usually produce low values for the intersection distances. Since we expect groups from the same object to usually have low intersection distances, situations which also produce low intersection distances when the groups come from different objects will not provide any useful information. In such cases, GROPER does not use the values of int_1 and int_2.

When GROPER does use int_1 and int_2, it does so such that lower values for these variables indicate an increased likelihood that the groups come from the same object. Furthermore, the discussion in chapter 3 suggests that this likelihood decreases linearly with the distance. For that reason, GROPER uses the following distributions:

$$P(int_i = x | O_1 = O_2) = \frac{2(MaxPartDiameter - x)}{MaxPartDiameter}$$

When determining $P(int_i = x | O_1 \neq O_2)$, GROPER uses a uniform distribution. Together, these distributions have the effect that as the distance increases to the place where the projection of groups intersect, the judged likelihood that the two groups come from the same object will decrease linearly.

5.3.4 Orientation, Different Objects

When considering the possibility that two groups come from different objects, GROPER decides on the chances that, when randomly oriented, those two groups would produce a $type_1$ orientation, a $type_2$ orientation or a $type_3$ orientation. In doing so, GROPER follows the distributions suggested in chapter 3.

To determine the probability that two groups would have a $type_1$ relationship, GROPER compares the angle of each group's projection to the angle which the other group presents to it, multiplying together the probability that group 1 falls into group 2's projection and the probability that group 2 falls into group 1's projection, as if they were independent.

GROPER determines the probability that group 2 falls in group 1's projection in two different ways, depending on whether group 1 has a finite or infinite projection.

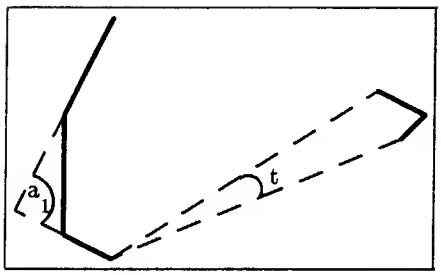

Figure 5.5: Two groups with a $type_1$ relationship. "t" represents the aspect the second group presents to the first group. "a_1" is the angle of projection of the first group. The probability that the second group falls in the first's projection is approximated by $\frac{a_1 - t}{2\pi}$.

If it has an infinite projection, GROPER subtracts the angle which group 2 presents to group 1 from the angle of group 1's projection. It calculates the angle group 2 presents by taking the angle formed by a line from the first point in group 2 to the first point in group 1, and a line from the last point in group 2 to the first point in group 1. Figure 5.5 illustrates this calculation. GROPER then divides the difference of these two angles by 2π. This roughly approximates the probability that group 2 would continue to be in group 1's projection if we rotated group 1 randomly, without altering its distance from group 2.

If group 1 has a finite projection GROPER does something similar. It first finds the point where group 1's first and last edges intersect. Then, it rotates group 2 around group 1 so that it falls inside groups 1's projection. It then measures the size of the face that group 2 presents to group 1 by taking the angle between a line from the start of the rotated group 2 and the intersection point, and a line from the end of the rotated group to the intersection point (see figure 5.6). Finally, it subtracts this angle from the angle of the projection, and divides by 2π. This approximates the probability that group 2 would fall in group 1's projection if we randomly rotated group 2 about group 1.

GROPER estimates the probability that two groups will have a $type_3$ relationship using the formula presented in chapter 3. That chapter presented a solution to the problem when the groups are small relative to the distance between them. When a_1 and a_2 represent the angles of the projection of the two groups, then:

$$P(type_3 | O_1 \neq O_2, rest of data) = \frac{\frac{3\pi}{2} - a_1 - a_2 + \frac{a_1 a_2}{2\pi}}{2\pi}$$

For this problem, if a group has a finite projection, GROPER just uses 0 as that group's angle of projection.

Finally, to determine the probability of $type_2$ occurring, GROPER just subtracts from

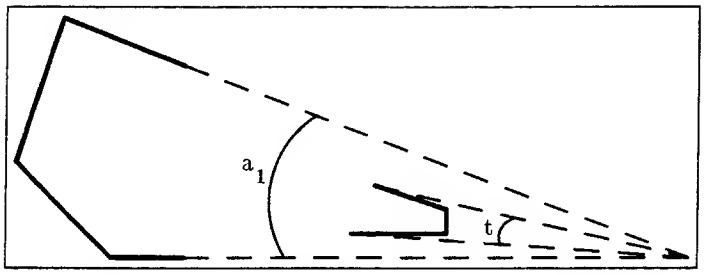

Figure 5.6: The group on the left has a finite projection. After rotating the other group to fall in this projection, "t" represents the aspect the second group presents, and "a_1" is the angle of projection of the first group. The probability that the second group falls in the first's projection is again approximated by $\frac{a_1-t}{2\pi}$.

1 the probability that $type_1$ or $type_3$ will occur. These probabilities, like most of the ones GROPER uses contain a certain amount of guesswork. We justify their use on two grounds. First of all, some real insight into the physical processes that produce images lies beneath this guesswork. And secondly, the grouping system that results from these guesses performs well. Chapter 9 will present examples of this performance.

5.4 Creating Larger Groups

Sometimes, after combining two different groups of edges, indexing will show that a large number of sets of edges from different objects might have produced all the edges belonging to the two groups. In such a case, we do not want GROPER to go back and find a different promising pair of groups. Instead, GROPER considers proposing for indexing all triples of groups which contain that pair of groups. It may propose one of these triples in two ways. First of all, it estimates the probability that all the edges in these three groups come from the same object. This allows it to compare these triples against the unexplored pairs of groups. But additionally, GROPER's grouping system tells its recognition system to explore five promising triples of groups before it considers any more pairs of groups. We have chosen the number five arbitrarily, any other small number might do as well. This prevents GROPER from nearly finding an object, and then wandering off to explore other interesting parts of the image.

GROPER estimates the probability that three or more groups of edges come from the same object by combining the probabilities that pairs of groups could have come from the same object. This allows it to compare triples of groups to each other and to pairs of

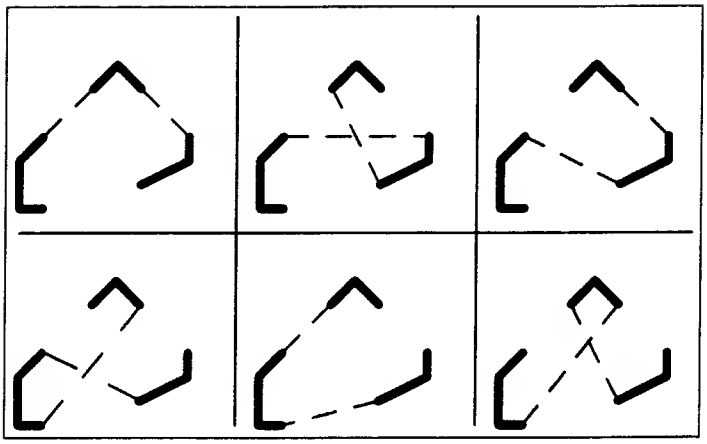

Figure 5.7: Six ways to combine three groups of edges. The thick lines represent groups, the thin dashed lines show how they connect.

groups. The probability that groups one, two, and three come from the same object is just the probability that groups one and two come from the same object, times the probability that groups two and three come from the same object, given that one and two do. We know how to estimate the probability that groups one and two come from the same object, we must consider how knowing whether these two groups come from the same object effects the likelihood that a third group comes from this object as well.

Recall from chapter 3 that our estimation of the likelihood that groups one and two come from the same object was really based on the likelihood that a path exists from one group to the other along the perimeter of an object, and that no intermediate section of perimeter has produced an edge in the image. We have six ways we can string these three groups together. Figure 5.7 illustrates these six possibilities. GROPER considers all six of these possibilities for each triple of groups it wants to consider, and picks the most likely ones to present to the indexing module.

GROPER breaks the problem up into these six parts, because it then assumes that to determine the likelihood that group one connects to group two, and group two connects to group three, it only needs to compute the probabilities of these two events separately, and then multiply them together. GROPER assumes that whether or not group one is closest to group two, and from the same object, does not effect the likelihood that group three is closest to the other end of group two. This assumption follows the whole spirit of this thesis which assumes that objects are in a sense random, and that what happens in one distant part of an object will not effect the likely appearance of another part of the

object. It would also be hard to see how we might use the information that group one and two come from the same object to influence our assessment of group three's likelihood of coming from that object.

So to determine the likelihood that group one connects to group two, which connects to group three, GROPER just multiplies together two probabilities. Calculating these two probabilities differs only slightly from the methods previously described to evaluate a pair of groups. In evaluating a pair of groups, GROPER used the minimum distance between the end of either group and the beginning of the other. In this case, GROPER knows that the end of group one is supposed to connect to the beginning of group two, and the end of group two is supposed to connect to the beginning of group three. So GROPER uses these distances. Other than that, GROPER calculates the two probabilities just as the previous section has described.

5.5 Summary

This chapter has explained how to build a grouping module based on the theory of grouping previously presented. In it we have had to make a few additional arbitrary decisions, for example, choosing the value of $\frac{1}{3}$ for the probability that two groups from the same object come from adjacent convex sections of that object, or choosing $\frac{1}{5}$ for the value used in arriving at the distribution of distances between groups from different objects. We did not use these values to fine-tune GROPER, however. In most cases they are the first values picked, based on guesswork and a rough idea of what was needed. Probably any similar values would result in a grouping system that performs as well. In spite of this guesswork, the grouping system described in this chapter relies mainly on the insights derived from the theory of grouping. The geometry of image formation has told us how to perform grouping.

5.6 Future Work

Two types of future work could improve the grouping algorithm presented in this chapter. First of all, at least two ways exist for improving the accuracy of this grouping system. Secondly, many ways exist for speeding up this algorithm. The existing implementation aimed mainly at testing the effectiveness of the evaluation metric at producing correct groups of edges, not in providing an efficient implementation. Efficiency was considered only when a slow system hindered development efforts.

GROPER's grouping system tends to make at least two kinds of errors which humans probably would not make when looking at images. First of all, GROPER will occasionally incorrectly combine some edges into the same convex section, hindering it from finding an

Figure 5.8: Dotted lines show straight line approximations to the edges found in an image. GROPER incorrectly combined the two solid edges into a single, convex section.

object which contains only some of those edges. Figure 5.8 illustrates an example of this. Secondly, GROPER's grouping system will often select a pair of groups which, in isolation seem to form a good group, but which do not seem to work well together when taken in the context of the entire image. Figure 5.9 illustrates this problem. Modifications to the grouping system aimed at solving these problems could improve its accuracy.

The first type of problem comes about because, although GROPER forms primitives out of the edges most likely to come from a single object, sometimes less likely interpretations are correct. GROPER tends to make two kinds of mistakes. First of all, sometimes the nearest edges did not actually come from the same object. Secondly, sometimes an object produces extended concave sections of edges. These problems occur rarely enough so that GROPER still works effectively in spite of them. But we do have a trade-off available. We could produce a more accurate system by producing more primitive groups or smaller ones, requiring more computation to find correct larger groups of edges.

We could handle nearby edges which do not come from the same object in two different ways. First of all, we could place stricter limits on the distance that we allow to separate two edges which GROPER will combine. Instead of always picking the most likely pairing of edges, we might only combine edges if they seem considerably more likely to go together than with any other edges. Often two edges have end points quite close to each other in an image, and a third edge has an end point also near these two. We might argue that in such cases, we have more than one quite likely way to combine edges, so we can not choose any with certainty. This would discourage us from making any questionable combinations of edges, producing more, smaller groups of edges about which we felt more certain. On the other hand, when confronted with several different options, all almost equally likely, we

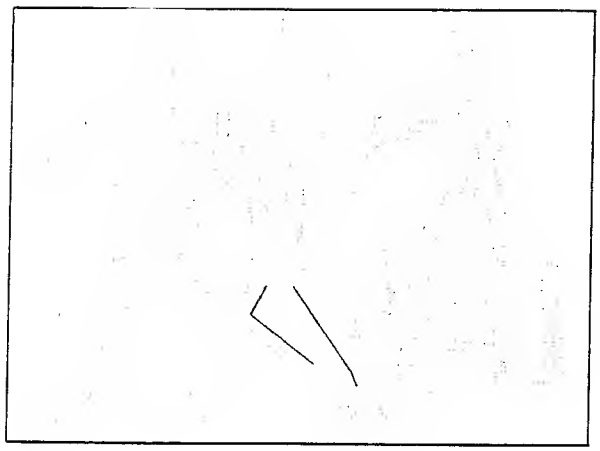

Figure 5.9: GROPER decided that the two groups shown go well together. While they might appear to go well together when looked at in isolation, people would not consider them a promising group when looking at the entire image.

might produce several different primitive groups of edges. This would increase the number of primitive groups, but still give us large groups with which to work. Either approach should produce a more accurate grouping system. When GROPER currently incorrectly combines two edges into the same group, it may lose the ability to use those edges to recognize an object. By creating smaller groups, or more alternate groups, this would happen less often.

A series of concave edges can also produce problems for GROPER. As figure 5.10 shows, a single concavity in an object's perimeter will not cause problems. Although GROPER will form an incorrect group out of the two line segments before and after this point of concavity, it will also form two correct groups which, when combined, account for the object's perimeter. However, a series of concavities, such as those produced by the gripping part of a wrench, will cause GROPER to form a single, large group, with figure and ground reversed. GROPER can then not use that group to correctly recognize an object, since the recognition module uses an incorrect surface normal for these edges. We might overcome this problem by creating a second, concave group out of every convex group. This would increase the amount of work needed to find good pairs of groups, but would avoid errors which occasionally crop up.

Reversing figure and ground can lead GROPER's grouping module to suggest groups that did not really all come from a single object. Such mistakes do not usually lead to the incorrect identification of objects, but they do waste a certain amount of GROPER's time. If an object has a point of concavity, it will consider the possibility that the two edges before and after this point form a convex pair pointing out from the object. When two

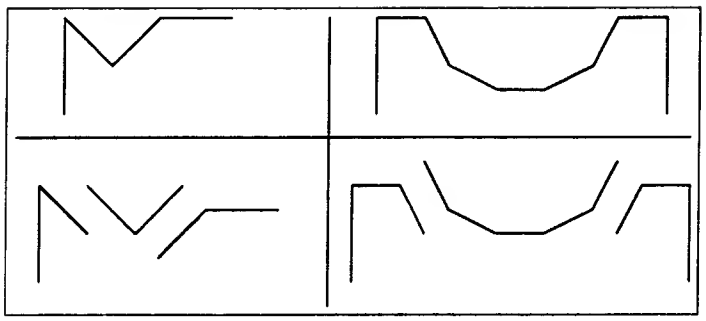

Figure 5.10: Two strings of image edges are on the top. The primitive convex groups they produce are on the bottom. On the left, even though one group has figure and ground reversed, all edges appear in a group which has the correct interpretation. On the right, three edges are lost to incorrect figure/ground judgments not made correctly in any group.

objects with concavities are nearby, only a short distance may separate two such groups. If they point at each other as well, GROPER will decide that they probably come from the same object. And, in fact, taken in isolation the edges do look promising. In the context of the entire image, these edges do not seem quite as good for two reasons. First of all, edges from each of the two groups will also usually group well with other edges from the object they really come from. So we might want to make the metric assigned to a pair of groups depend also on how well the edges in those groups go with other groups. Secondly, a pair of groups from nearby objects, with figure and ground reversed, will usually not go well with other groups of edges from either object because the other groups will usually have figure and ground assigned correctly. So, we might also want GROPER to judge a pair of groups partly on how well we could extend that pair with an additional group. These two approaches should improve GROPER's performance at telling figure from ground, without requiring a pre-processing step to determine this.

Two straightforward methods could also increase GROPER's speed. The theory of computation of grouping provides the location of groups likely to go well with a particular group. We know that groups near a particular group, or in its projection will tend to have the highest evaluation metric. Groups in $type_1$ can have high metrics even when far apart, because they point at each other in an unlikely way. Consider, for example, the two ends of a long rectangle. But distant groups with a $type_2$ or $type_3$ orientation cannot have high metrics. Yet GROPER spends most of its time evaluating the metric for every pair of groups. Instead, GROPER could start by looking at nearby groups, and groups which fall inside each others projections. Only if no such pairs of groups receive high metrics would

GROPER need to continue looking at the less likely candidates. By ordering the search for good pairs of groups, GROPER could avoid ever needing to evaluate the metric for most pairs of groups.

The grouping problem also lends itself naturally to a parallel solution. Most obviously, GROPER could find the metrics that describe each pair of groups in parallel, since this operation contains no dependencies. Obviously, implementing good parallel algorithms for recognition requires much more work than this. Parallel vision algorithms have had their best success on problems which can be solved by analyzing only local sections of an image, separately, although Mahoney[19] describes some work on parallel implementations of higher level processes. While not as naturally parallelizable as edge detection, we just wish to suggest that grouping, too, has an inherently local character which makes it an attractive candidate for parallel implementations.

Chapter 6

Indexing

Once GROPER has collected a likely group of image edges, it needs to determine which, if any, object's edges might have produced those image edges. GROPER does this by determining some properties of that group of image edges, and then looking up in a table to find modeled object edges that might produce image edges having those properties. This approach relieves GROPER of the necessity of explicitly considering every possible combination of object edges. Instead, with indexing, GROPER only needs to consider combinations of edges that have the right properties.

GROPER's indexing module uses as input, in a pre-processing step, modeled object edges, and in a run-time step, a group of image edges. GROPER obtains the object edges from a human operator, who provides lists of straight line segments that approximate the perimeters of the objects for which GROPER will look. Along with these line segments comes information about their outward-pointing normals, that is, on which side of the line segment the object lies. Furthermore, a human operator provides GROPER with estimates of the maximum likely error that will later occur in sensing these edges. Error can occur in the location or the orientation of an edge. The run-time input also consists of a list of line segments and their normals, this time provided by GROPER's grouping module, as described in the previous chapter. GROPER uses the pre-processing input to build up a table describing the relationships between pairs of edges, it uses the run-time input to look up suitable combinations of edges in this table.

When given a group of image edges, GROPER's indexing module responds with lists of line segments from object models that might have produced these edges, allowing for some error, or partial occlusion.

We have priorities on the way GROPER performs this task. We want to minimize the mistakes it makes, of course, but if indexing turns up some groups of object edges that could not have produced the group of image edges in question, this will not hurt

GROPER's performance as much as missing some correct matches. If indexing failed to produce a correct match, GROPER would have no way to recover from such an error, and would fail to recognize an object that produced that group of image edges, unless that object produced another set of edges that GROPER's grouping system could turn up later. It does not hurt GROPER's performance nearly as much if its indexing component turns up some groups of object edges that could not really have produced the image edges in question, because the subsequent verification step will weed out these mistakes. Such mistakes will waste some time by causing GROPER to attempt to verify them, but these attempts should fail. GROPER's accuracy may also suffer a little from such mistakes, because its decision as to whether to perform verification on the results of indexing will depend on how many matches indexing turns up. If indexing turns up extra, incorrect matches, GROPER will make a less well-informed decision about whether to try to verify the results of indexing. But producing incorrect matches with indexing will less often cause GROPER to waste time or miss an object than will failing to produce correct matches.

Also, GROPER will require pre-processing space and time to build the tables used in indexing. But the time required to build up a table for indexing will not concern us much compared to the time required to perform indexing steps at run time. We must require that GROPER use only a reasonable amount of space for this table, however. If the entire table can not fit in primary memory, this will slow down run-time performance. So, GROPER's indexing focuses on minimizing the amount of processing required at run-time, and avoiding missing possibly correct matches.

6.1 GROPER's General Approach to Indexing

Because GROPER makes use of straight line approximations to curved surfaces, it can use a method of indexing that takes advantage of simple descriptions of groups of edges. We can precisely describe the relationship between a group of straight lines with a limited number of parameters. For example, we can fully describe two lines with five parameters. We can then build a five dimensional table using the object models, with an entry for each set of five parameters that two edges of the object might produce. When given a pair of image edges, we would then just need to compute the parameters that describe them, and look up in the table to find all pairs of edges that could produce those five parameters. This brief description avoids some complications we must deal with, but GROPER's indexing does not differ too much from this simple approach.

Some past work has also been done on this subject. Schwartz and Sharir[22] have dealt with the complexities that ensue when we wish to index using continuous curved edges, instead of straight lines. They do not, however, apply this approach to unconnected

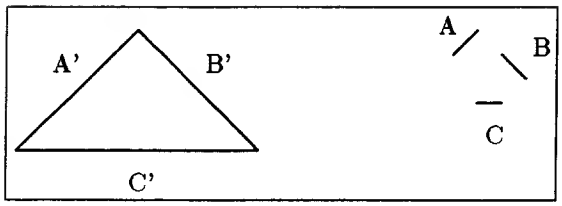

Figure 6.1: On the left, three edges from a model. On the right three image edges. Each pair of image edges is compatible with the corresponding model edges, but as a whole, the image edges do not match the model.

curved edges. Wallace[25] develops an indexing system for pairs of unconnected edges, but does not deal with error or occlusion. Thompson and Mundy[23] use indexing to find the pairs of vertices in a three-dimensional object that correspond to a pair of vertices in its two-dimensional image. This type of approach cannot handle arbitrary groups of image edges, but it does address the problem of recognizing three-dimensional objects. Another important approach to indexing is Ettinger's parts-based approach (Ettinger[7]). His system looks for common sub-parts of objects, and uses them to index into a library of objects. Chapter 8 discusses these approaches in more detail.

The main piece missing from the simple description above is that we wish to index using groups with more than two edges in them. Although we could simply compute more parameters describing groups containing more edges, it would not be practical to build up a table that contains entries for the parameters that describe every possible combination of edges in an object; if an object has n edges it can produce 2^n different combinations of edges.

So instead, GROPER breaks the indexing problem into smaller problems and combines the results. It builds a table that contains entries for every pair of edges in each object model. Then, to identify the model edges that might have produced a group of image edges, GROPER looks in this table to find all the pairs of edges that might have produced each pair of edges in the group and combines the results. This approach leaves two separate problems: how to perform indexing with a pair of edges, and how to combine the results.

In determining how to perform indexing, we must decide on a particular parameterization of the relationship between two edges. Furthermore, we must keep in mind that, when building the lookup table, we need to make entries, not just for the particular relationship between two model edges. We must consider all possible relationships between the image edges these model edges can produce, considering error and the fact that only a portion of a model edge may show up in the image.

In combining the results of these lookups, we must consider the particular problem that a set of pairwise consistent matches may not be globally consistent. For example, figure 6.1 shows three image edges, A, B, and C, and three modeled object edges, A', B', and C', which form a triangle. Clearly A, B, and C could not have come from model edges A', B', and C' respectively. And yet, upon examination we see that edges A' and B', considered alone, could have produced edges A and B. Similarly, B' and C', considered alone, might have produced B and C, and A' and C' might have produced A and C. GROPER's indexing system therefore takes special care in order to come up with matches between image edges and object edges that have global consistency. Grimson and Lozano-Pérez[10] discuss this issue in more detail, and provide some information about how frequently it occurs.

6.2 Choosing a Parameterization

Many different ways exist of choosing parameters that characterize the relationship between two edges. A good choice will have a number of desirable traits. First of all, we would like a parameterization that we can compute quickly, at run time. Secondly, we would like a parameterization that will help us to ensure the global consistency of matches when we combine the matches found for pairs of edges. As this section will relate, GROPER has a parameterization particularly suited to this task. Thirdly, a good parameterization will facilitate the construction of an object library by making it easy to find the range of values a pair of edges may create, when subject to error and occlusion.

This problem of local consistency and global inconsistency emerges because the fact that edges A, and B could come from some fragment of edges A' and B' does not tell us which fragments of A' and B' could produce A and B. Does A come from the beginning of edge A', its end, or somewhere in the middle? The match of A and B to A' and B' may work only if A comes from the beginning of A', while the match of A and C to A' and C' works only when A comes from the end part of A'. We can check these two local matches for consistency by seeing what they imply about the location of the fragment of A' that produced A, relative to the start of A', and making sure these locations are compatible. So, GROPER uses a parameterization of edge relations that makes it easy to tell, when a pair of edges match modeled edges, what this match implies about the location of one of the image edges in the model edge that produced it.

GROPER makes use of two different parameterizations, one for nearly parallel edges, and the other for non-parallel edges.

6.2.1 Non-parallel Edges

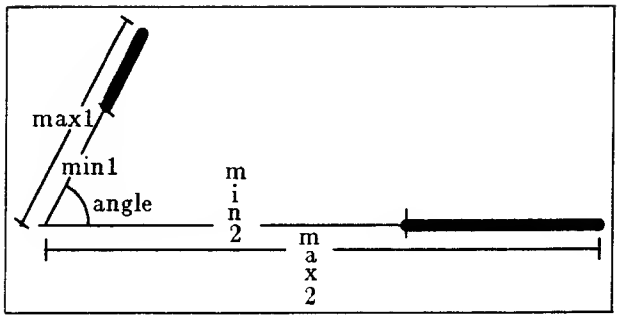

Figure 6.2: Two edges are in bold. Five parameters describe their relationship.

GROPER uses the angle between two edges as one of the parameters that describes them. GROPER then calculates the point in the plane where the two lines defined by the edges intersect. GROPER then measures the minimum and maximum distances from each edge to this point. Figure 6.2 depicts these five parameters.

These parameters will make it easy to discover at what part of an edge an edge fragment originates. For example, suppose we have two perpendicular model edges located at ((0 10) (0 20)) and ((15 0) (23 0)). We would describe these edges with the angle $\frac{\pi}{2}$, and the distance ranges (10,20) and (15,23) respectively, since the two lines the edges define would intersect at the origin. If we then detected two image edges with parameters $\frac{\pi}{2}$, (15,20) and (15,23), we would know that the first image edge must come from the second half of the first model edge.

With this parameterization it also becomes easy to determine the range of parameter values that fragments of two edges might produce. Two fragments of edges always appear at the same angle as the original edges. And since a fragment of an edge defines the same line in the plane as the full edge, two fragments of edges have the same intersection point as the two original edges. So the maximum and minimum distance from the edge fragments to the intersection point will fall within the range of the maximum and minimum distances for the whole edge.

Error can also affect the range of possible values for these parameters. If error occurs in detecting the angle between two edges, this will obviously affect the angle between the edges, but it will also affect the distance from the edges to their intersection point. Suppose an error of $\frac{\pi}{10}$ may occur in determining the angle of an edge. Then, for the two edges described above, instead of always expecting an angle of $\frac{\pi}{4}$ between the image edges they produce, we must expect an angle somewhere in the range $(\frac{3\pi}{20}, \frac{7\pi}{20})$. Furthermore, error in angle can also effect the location of the intersection point for two edges. For example, if the first edge produces an image edge still sensed to begin at (0 10), but with a $\frac{\pi}{10}$ difference in

its angle, the intersection point between this edge and the edge from (15 0) to (23 0) may fall anywhere between $(-3\frac{1}{4},0)$ and $(3\frac{1}{4},0)$. This error alone will affect the minimum and maximum distance from each edge to the intersection point, and of course error may alter the sensed angle of the other edge as well. The extent to which error in angle affects the maximum and minimum distances to the intersection point will depend on the initial angle between the edges, as well as on their location. For example, error in the angle between two almost parallel edges will affect the location of their intersection point far more than it will affect two perpendicular edges. When GROPER makes entries for two edges in its indexing lookup table, it must make sure that these entries cover all possible parameters these edges can produce; and these parameters can vary quite a bit, in related ways, due to error in the sensed angle of an edge.

In order to build the indexing table, GROPER calculates the ranges that the four distance parameters can assume for a particular range of angle values. GROPER finds the extreme possible distance values for this range of angle values simply by calculating the distance values for the extreme possible orientations. First of all, GROPER considers that each edge might produce a tiny fragment, either where the edge starts or where the edge ends. The two edges could produce any pair of such fragments, one from each edge. The extreme distance values come from these pairs of fragments. We consider only tiny fragments, because, at this stage we calculate only the effect of errors in angle. Taking a full edge and rotating it by $\frac{\pi}{10}$ would produce not only changes in the angle of the edges, but also large changes in the location of the edge. Upcoming paragraphs will discuss how GROPER separately accounts for the effects of errors in the location of the edges.

So GROPER considers all four possible pairs of these extreme edge fragments. Then for each pair, we consider four different possible orientations of that pair. The range of allowed angles provides a minimum and maximum possible angle. And for each of these extreme angles we consider tilting the edges clockwise as much as possible, while still achieving that angle and staying within the maximum allowed error in orientation of an edge. And we consider tilting the edges as much as possible counter-clockwise. So we have four possible extreme pairs of edges, for each pair we can tilt the edges to produce the minimum or maximum allowed angle, and then for each of these eight combinations we can tilt the edges as much as possible clockwise, or as much as possible counterclockwise, producing a total of sixteen possible combinations. GROPER computes the maximum and minimum distance to the intersection point produced by each combination, and then takes the total maximum and minimum values over all these combinations as the values of the needed parameters, accounting for angle error.

So we find the extreme values by only considering the extreme angles, and the edge fragments at the ends of the edges. To see that extreme values do not occur in other places,

notice that as we vary from one extreme to another, the parameters produced also vary from one extreme to another. For example, if we varied the angle between the extreme minimum angle and the extreme maximum angle, we would be able to rotate the edges more and more, producing values that change continually in one direction only.

Sensing edges can also produce error in determining the location of the edge. We can divide this error into two parts. Error may displace the location of an edge in a direction tangent to the edge. This will result in error in determining the maximum and minimum distance from that edge to the intersection point. For example, if we sense the above mentioned edge at ((0 12) (0 22)) instead of at ((0 10) (0 20)) then we will find it has minimum and maximum distances to the intersection of (12, 22) instead of (10, 20). But this will not effect the distance between the other edge and the intersection point. Alternately, error in the direction normal to an edge will produce a change in the distance from that edge to the intersection point. So the effect on these parameters of error in the location of an edge depends on the angle between the two edges. Potentially, a small error in location can cause a big change in the value of these parameters, if the two edges form a small angle. In such a case, a small error in determining the angle of an edge will also produce a big change in the location of the edges' intersection point, and hence in their maximum and mininum distances to that point. So we can approximate the effect of error in the location of an edge by adding a small amount of padding to its maximum and minimum parameter values. In cases where a small error in the location of an edge results in a big change in these values, we will have already put a lot of padding into the allowed range of these values, by accounting for the large possible changes caused by possible errors in the angles of the edges. In this way, determining the possible values of these parameters in the face of error becomes a problem of analyzing the effect of error in the angle of the edges. As the confusing nature of this discussion indicates, this particular choice of parameterizations does not lend itself especially well to determining the ranges of parameter values in the presence of error.

Once we know how to compute the range of possible parameters for a pair of edges, we can build a table of all possible parameters for all pairs of edges. Two problems emerge when we do this. First of all, we must quantize these values. And secondly, every pair of edges has quite a large number of possible parameter values.

In order to put values into a table we need to round them off. This quantization will introduce some extra padding in the range of allowed parameter values. To determine the table entries we need to make for a specific pair of edges, GROPER considers the range of angles these edges might produce, given the tolerated amount of error in sensing the angles between them. Then GROPER divides this range up into subranges. GROPER uses subranges of size $\frac{\pi}{10}$. A pair of edges might produce angles in a range from $\frac{1.3*\pi}{10}$ to

$\frac{3.3*\pi}{10}$. GROPER divides these into the subranges $(\frac{1*\pi}{10}, \frac{2*\pi}{10})$,$(\frac{2*\pi}{10}, \frac{3*\pi}{10})$, and $(\frac{3*\pi}{10}, \frac{4*\pi}{10})$. Then for each subrange, GROPER determines the minimum and maximum possible distance from each edge to the intersection of the edges, using the algorithm described above. GROPER adds some extra padding for possible errors in the locations of the edges, and then uses these values to make entries in the table.

We could view our table as a five dimensional array, containing an entry for every possible quintuple of parameters an edge might produce. This approach has the disadvantage that a single pair of edges would produce a large number of table entries. We must consider every combination of distance ranges, one from each edge. And each edge will have many valid distance ranges, one for each subsection of the edge. So each edge has a number of distance ranges proportional to the square of its length. And the number of different pairs of distance ranges for a given angle will be proportional to the product of the squares of the edge lengths.

Instead, a trick reduces the amount of space needed, at the cost of some run time processing when GROPER performs lookups. Instead of a five dimensional array, GROPER uses a three dimensional array, indexed with the allowed angle between the edges, the allowed distance from edge one to the intersection point, and the allowed distance from edge two. So GROPER makes an entry in the table for every allowable distance from a point in edge one to the intersection, and from a point in edge two to the intersection. Later, when performing a lookup based on two image edges, GROPER actually takes the intersection of two lookups. First it looks in the table under the angle between the edges, and their minimum distances to the intersection point. Any pair of model edges that might have produced these two image edges will have an entry there. Then GROPER does a second lookup, using the angle between the edges and their maximum distances. The intersection of these two lookups provides all pairs of model edges that could match these image edges. This requires extra time, both because of the two lookups required, and because GROPER must intersect two sets, potentially a good deal larger than the result. But the table entries for a pair of edges now only require space proportional to the product of the lengths of the edges.

This explains how GROPER finds the pairs of object edges that match two image edges. It remains to discuss how GROPER combines these pairs to find the consistent matches for a group of edges. It does this by keeping track, for each match, of the possible amount of the front of one of the model edges that might be missing from the corresponding image edge. Suppose, for example, GROPER finds a match between image edges A and B and model edges A' and B'. And suppose GROPER knows that, even with error, model edge A' can not have a point less than 20 units away from A's intersection point with B'. Then, if A begins 20 units away from its intersection with B, GROPER can conclude that the

beginning of A is also the beginning of A'; none of the front of A' has failed to show up in A.

In general GROPER does this in the following way. When making an entry in the lookup table for a pair of edges, GROPER records the minimum and maximum possible distance between the start of the first edge and the intersection point of the two edges. GROPER also records the length of the first edge. Call these three values min-d, max-d, and obj-len, respectively. When looking up a pair of edges that matches this pair, GROPER will know the minimum distance between the first edge in the image and the intersection point, as well as the length of the first edge. We will refer to these values as min-im and im-len. GROPER can now find the minimum possible length of any missing front part of the model edge with the expression:

(max 0 (- min-im max-d))

On the one hand, the length of any missing front section of the edge can not be less then 0. On the other hand, this length also can not be less than the difference between the distance from the start of the image edge and the intersection point, and the largest possible such distance. For example, if the image edge starts 40 units from the intersection point, and the largest possible distance from the start of the model edge, considering error, is 30 units, then at least the first 10 units of the object edge must be missing from the image edge.

Similarly GROPER calculates the largest possible length that could be missing from the front of the image edge with the expression:

(min (- min-im min-d) (- obj-len im-len))

This distance can not exceed the difference in lengths between the two edges. It also can not exceed the difference between the distance from the start of the image edge to the intersection point, and the smallest possible such distance for the object edge.

GROPER then combines these results to check the consistency of a match between a group of image edges and a collection of object edges. Suppose GROPER wants to find the model edges that might match the image edges (A B C D). GROPER pairs edge A with each of the three other image edges, and performs a table lookup for each pair. With each match to a pair of model edges, GROPER determines a range of possible amounts missing from the front of A. Then when GROPER combines some matches, it intersects these ranges. If the intersection is null, GROPER knows that those matches are not compatible. GROPER finds the matches for (A B C D) by repeating this process using B, C and D in turn as the first edge, pairing each with the remaining three edges. GROPER finds the sets of model edges that match these image edges and provide a consistent set of values for the amount missing from the first image edge. It takes the intersection of all these sets of matches,

producing the matches for which a consistent location exists for all the image edges with respect to the corresponding model edges.

This approach requires that GROPER combine the results of performing a separate table lookup for every ordered pair of edges in a group. Furthermore, for each pair GROPER actually takes the intersection of two lookups, using first the minimum and then the maximum distances to the intersection point. At the cost of some extra run-time processing, this provides greater accuracy of indexing, while reducing space requirements.

6.2.2 Parallel Edges

The parameterization discussed in the previous section is not suitable for parallel or nearly parallel edges. Parallel edges have no intersection point. So if we allow for error, nearly parallel edges will produce an infinite range of possible distances to an intersection point. This section discusses a second parameterization used for almost parallel edges.

When making table entries for a pair of edges, GROPER does not decide to exclusively use one set of parameters or the other. Rather, for errors in the angle of the edges that would make them nearly parallel, GROPER calculates the possible values of the parameterization discussed in this section. For errors in the angles of the edges that do not make them nearly parallel, GROPER uses the parameterization discussed in the previous section. For example, suppose two edges have an angle of $\frac{1.5\pi}{10}$. GROPER considers two edges nearly parallel if they are within an angle of $\frac{\pi}{10}$ of being parallel. However, allowing for error, these two edges may form an angle anywhere from $\frac{2.5\pi}{10}$ to $\frac{.5\pi}{10}$. GROPER breaks this range up into three subranges: $(\frac{3\pi}{10},\frac{2\pi}{10})$, $(\frac{2\pi}{10},\frac{1\pi}{10})$, and $(\frac{1\pi}{10},0)$. For the first two subranges, GROPER makes entries using the non-parallel parameterization of the previous chapter. But for the last subrange, GROPER calculates the possible values that a different set of parameters may have, as this section will describe.

For nearly parallel edges, GROPER uses the following five parameters: the angle between the edges, the minimum and maximum distance from the middle of a fragment of the first edge to any point on the second edge in the direction tangent to the first edge, and the minimum and maximum distances in the direction normal to the first edge. These five parameters do not fully specify the relationship between the two edges, because they do not indicate the length of the first edge. However, this choice of parameters does allow GROPER to fit these five parameters into a three dimensional array, just as it did with the parameters for non-parallel edges. Furthermore, since GROPER puts the length of the first edge into the table with each entry, it can confirm that no matched pair of image edges has a first edge longer than the matching first model edge (taking possible error into account).

GROPER calculates the range of possible values for these parameters in much the

same way as it did for the non-parallel parameterization. It also uses a similar method for enforcing the global consistency of matches. However, parallel image edges do not always provide much information about the location of the image edge within the object edge. If, for example, we have two parallel model edges, ((0 0) (0 10)) and ((5 0) (5 10)) that produce two image edges ((17 1) (18 1)) and ((17 6) (18 6)), we can not tell anything about what part of the model edges the image edge fragments come from.

To enforce what consistency it can, GROPER stores in an entry the minimum and maximum possible distance from the midpoint of a fragment of the first edge to the second edge, in the direction of the tangent of the first edge, along with the length of the first edge. GROPER then applies exactly the same procedure as it used for non-parallel edges.

Thus the parameters chosen for almost parallel edges allow GROPER to build and use the index table in the same way for parallel and non-parallel edges. In both cases GROPER uses the same three dimensional array, combining the results of two separate lookups. This is possible because, in both cases, four of the parameters actually represent two ranges of possible values, which may vary when we take fragments of the model edges. GROPER can also use the same procedure to enforce consistency in both cases, because each parameterization uses a value that varies according to the length of the section of the front of the model edge that fails to show up in the corresponding image edge. However, because the parameters discussed in this section do not fully describe the relationship between the two edges, some inefficiency may result. The initial table lookup may produce some pairs of edges in which the first edge is too short to match the first image edge. Although GROPER weeds these out using the length of the model edge as stored in the table entry, this process may take some extra time. A more complete parameterization might allow GROPER to never have to consider these matches at all.

6.2.3 Evaluation

First of all, indexing will produce incorrect matches because it makes many simplifications. For example, GROPER puts table entries into buckets that cover a significant range of values. So GROPER may find that a pair of image edges produce a parameter value of 1, and use that value to look up pairs of model edges in a table. If buckets have a length of 20, then in the same bucket of the table that has the correct pairs, GROPER may also find a pair of edges that could not produce a parameter value of less than 19. Mistakes may also creep in because GROPER, for a particular range of angle values, calculates the maximum and minimum values that a pair of edges might produce for the other parameters. However, while a pair of edges with an angle between them of $\frac{\pi}{2}$ might produce a value of 10 for one parameter and a value of 20 for a different parameter, it might not be able to produce both

values at the same time. Yet the table entry will not indicate that. Chapter 9 will discuss tests that show how frequently these errors occur, and argue that they will not have much effect on GROPER's overall performance.

Secondly, as mentioned before, GROPER's approach to indexing seeks to simplify the enforcement of global consistency, and the building of a three-dimensional array that describes all possible relations between fragments of edges. It does this at the expense of some ease of implementation. But more importantly, this approach to indexing requires combining the results of a large number of table lookups performed on a fairly densely packed table. This results in greater run-time costs.

The amount of time GROPER takes to perform indexing with this scheme will depend mainly on the density of the lookup table. To find matches for a group of image edges, GROPER performs the following pseudo common lisp code fragment:

```
(let ((consistent-matches-list                    ; list of matches with
       (loop for edge in group-of-edges           ; consistent location
             collecting                           ; for a single edge.
           (let ((matched-pairs
                  (loop for edge2 in (remove edge group-of-edges)
                        collecting
                        (intersect (lookup-min-values edge edge2)
                                   (lookup-max-values edge edge2)))))
             (combine-into-consistent-matches matched-pairs)))))
  (intersect-matches consistent-matches-list))
```

The function "intersect" takes the intersection of the matches found in the table for the minimum parameter values a pair of image edges produces, and for the maximum values. If a group has M image edges, GROPER must perform this function $M(M-1)$ times, with a cost that increases linearly with the number of values in an average cell looked up. "Combine-into-consistent-matches" takes all the pairs of model edges that match the image edge pairs and, for one of the image edges, constructs all groups of matches that imply a consistent location for that one image edge. GROPER must perform this M times, with the cost of each related to M times the number of matches for a typical pair of image edges. Finally, "intersect-matches" intersects the M results of this routine, with a cost proportional to the number of sets of model edges that consistently match the image edges for the position of at least one of the image edges.

What does this analysis tell us? First of all, we can measure the cost of using a three

dimensional array instead of a five dimensional array as $(M)(M-1)$ times the typical number of entries in one cell of the array. However, as the number of objects in the model library grows, we expect this density to grow linearly. This means that this indexing scheme will have an $O(M^2N)$ cost, for N objects modeled in the library. Chapter 9 provides more specific information about experiments that measured the density of the lookup tables.

Secondly, we see that the way to reduce the cost of indexing in general is to reduce the density of the lookup table. Most of the costs of indexing depend on how many matches turn up at each stage. The cost of "intersect" depends on the number of entries in a cell. The cost of "combine-into-consistent-matches" depends on the number of model pairs that match the image pairs. Only "intersect-matches" depends mainly on the number of groups of model edges that will match the group of image edges in question. With accurate grouping, a recognition system should not need to perform too many indexing steps. Nonetheless, for large libraries the indexing described in this chapter may prove too slow. The following section describes some ways to speed up indexing by using a more sparse lookup table, at the expense of some space.

6.3 Alternate Approaches to Indexing

In indexing, a fundamental trade-off exists between space and time. To speed up indexing we would like to use a less dense lookup table, and require fewer steps to combine the results of indexing. We can accomplish both these goals by performing indexing based on triples of edges instead of pairs. A straightforward implementation of this idea will require a lot more space, however, so we will also suggest some ways to address this space problem.

Suppose we built a lookup table with entries that describe the possible relationships between every triple of model edges. Such a table would have two key advantages. First of all, it would remove the need to perform extra lookups to check the global consistency of matches. Secondly, it would create a much sparser table.

When a pair of non-parallel image edges match a pair of model edges, this match determines the rotation and translation needed to bring the model into alignment with the object in the image. Global inconsistency arose with matches between a series of pairs of edges because the rotation and translation implied by one match might differ from the one implied by another match. Using triples of edges instead of pairs, however, we can insure that each triple has two edges in common with one other triple. If these two edges match the same two model edges in a match for each triple, then the two matches must actually require the same rotation and translation, implying a global consistency between all the edges in the two matches.

An example will make this clearer. Suppose we have a group of image edges, A, B, C,

D. We use the parameters that describe edges A, B, and C to look up in a table, finding the matching model edges A', B', and C'. A second lookup reveals that B', C', and D' match B, C, and D. In both matches, B and C are matched to B' and C'. Matching B and C to B' and C' fully specifies the location of the object in the image. So this location must be the same for the first match as it is for the second. And the location of the object that aligns A with A' must also align D with D'. In contrast, when we used pairs of edges, A and B might match A' and B' for one location of the object, while B and C matched B' and C' for a completely different location of the object. So those two matches between pairs of edges would tell us nothing about the compatibility of A matching A' with the match between C and C'.

Some potential problems still exist for this approach, which has not been implemented. Because a match between two parallel image edges and two parallel model edges do not fully determine the location of the model in the image, we should order these lookups so that no two triples have only two parallel edges in common. And a single lookup specifies the location of the object model only to within some error bounds. Over multiple lookups these errors might accumulate, so that the location of the object implied by the first match differs slightly from the location implied by the second match, which again differs a bit from the location implied by the third match. This might result in the first and third matches differing by more than we would like to allow. However, it seems probable that we can implement an indexing scheme using edge triples, which will eliminate most problems of global inconsistency. And, while GROPER's indexing approach requires that to find the matches for a group of N edges it must find the matches for $(N)(N-1)$ pairs of edges, using triples it would only need to consider $N-2$ sets of edges.

Edge triples have the second advantage of creating a sparser lookup table. So that we may easily compare the sparseness obtained by using triples with that obtained with pairs, let us forget about the trick that allows GROPER to use a three-dimensional table, and suppose that a single lookup will produce only model edges that could have produced the two or three image edges. This means using a five dimensional array for pairs of edges, and a nine dimensional array for triples. (Nine parameters suffice to capture the relationship between three edges. We could use two angles, the location of the centers of two edges relative to the center of the third, and the lengths of the three edges). The run-time cost of each indexing scheme will depend on how many matches a single lookup produces, since we must then take the intersection of a series of such lookups.

Our library of object models will typically have fewer triples of edges that match three image edges than pairs that match pairs of image edges. To argue this, we may think of a triple as consisting of a pair of edges plus one extra. We then consider whether adding the extra edges creates more matches than the pair alone had, or fewer. For any pair of

model edges that match the first two image edges, we will usually find at most one model edge that could have produced the third image edge, and often we will find none. This is because matching the first two image edges will determine a specific location of the object model in the image, unless the edges are parallel. Given that single possible location of the object model, only one model edge will align with the remaining image edge. Allowing for error may occasionally produce more than one possible match. But on the other hand, if the first two image edges did not really come from an object that they happen to match, we can expect that quite often that object will not have any match for a third image edge. Chapter 9 shows some experiments that indicate the amount of a reduction in processing we could achieve using edge triples instead of doubles.

The use of edge triples instead of pairs offers the potential for significant speed-ups in the time required for indexing. However, it would also require significantly more preprocessing time and space. While a model with N edges has only $O(N^2)$ pairs of edges, it has $O(N^3)$ triples of edges. This might make the use of triples counterproductive when N grows large enough so that the lookup table fills available primary memory. However, we have a variety of possible compromises at our disposal.

We could use two lookup tables, one for pairs of edges and one for triples. This allows us to save space if we put every pair of edges in the first table, but only some of the triples in the second. This will work as long as we can tell at run time in which table to look.

For example, because grouping tends to lump together edges that come from single convex sections of an object, we could make entries in the triples table for all triples of edges for which two of the edges come from the same convex section of the object. Then when we form a group of image edges that appear to have two edges from one convex section of an object, we could look up all the triples in this group in the triples table, not using the pairs table at all. We would only need to use that table if we had a group of edges for which all edges seemed to come from convex sections that produced no other edges in the image.

Alternately, since grouping favors nearby edges, we could make entries in the triple table when two, or maybe all three of the edges could produce fragments sufficiently close to each other. Then, when looking up a group with edges near each other we could use the triples table, using the pairs table otherwise.

In a sense Lowe's system SCERPO [18] has already used the idea of creating special lookup tables geared towards the characteristics of a grouping system. It performs grouping based on certain properties, such as parallelism or co-termination. Then SCERPO indexes into a table containing only those edges that have these special characteristics. For example, if SCERPO finds two parallel edges, it performs an indexing step to produce all pairs of parallel edges in the object model for which it is looking. This immediately produces a set

of possible matches without search.

The reader can imagine ever more elaborate possible indexing schemes, including schemes that use even more than three edges when making entries in a lookup table. However, the serious investigation of these possibilities, if it happens, should be driven by the requirements that emerge as researchers investigate more complex recognition problems with larger object libraries.

Chapter 7

Verification

When grouping and indexing have found a collection of image edges that could only match a limited number of sets of model edges, GROPER uses verification to decide which if any of these matches to accept. The verification process takes a match between image edges and model edges, and determines a rotation and translation that will bring the object model into alignment with the image edges when placed in a common coordinate system. If no such rotation and translation exist, then GROPER knows that this supposed match does not really work. If GROPER does find a rotation and translation, it uses them to look for other image edges that can align with edges of the model. Finally, based on all the matching image edges, GROPER decides whether the image provides enough support for this hypothesized position of an object. If it does, GROPER accepts the hypothesis and removes all matched edges from the image before looking for more objects.

This process presents three problems. First, how to find a rotation and translation based on some matches between edges. Second, how to look for more edges based on this rotation and translation. And third, how to decide if a match provides enough evidence for the existence of an object. GROPER does not do anything original in addressing these questions. So we will discuss these questions only briefly, referring the reader to previous work for more detailed information. However, we must also consider two other issues. Before describing how GROPER performs verification, we explain why GROPER needs verification when it already has an indexing module. And after describing the verification module, we evaluate it in light of the kinds of problems that prompted the development of GROPER in the first place.

7.1 Why Verification?

GROPER needs a verification module for two reasons. First of all, as chapter 6 explains, indexing may make mistakes, and come up with some sets of model edges that could not really have produced the image edges used to index. Secondly, a hypothesis driven verification step offers a more efficient method of finding additional support for a match then does further grouping and indexing.

Finding a correct correspondence between some image edges and modeled object edges does not completely solve GROPER's recognition problem. If GROPER finds that a group of image edges could match a few different sets of model edges, GROPER still does not know which of the matches is correct. Furthermore, GROPER must consider the alternate hypothesis that the image edges do not all really come from a single, known object, but just coincidentally match up with a few known objects. Before having to decide between these alternatives GROPER would benefit by knowing the additional edges in the image for which any of the proposed matches might account. If GROPER finds that one of the proposed matches would also account for several other edges in the image, while the other matches derive no additional support, this might lead GROPER to prefer the first match over the others. Also, once GROPER succeeds in recognizing an object, it needs to remove from the image all the edges that object produced, so that it does not use any of them in recognizing different objects.

We could imagine looking for additional edges that fit our matches using grouping, but that would not produce a quick and certain answer. Even if some edges do not combine well with the image edges in the current group, we still want GROPER to find them if they would align with one of the matched objects. And once we have a small number of hypothesized matches, it is easy to see where those matched objects would appear in the image, and compare their proposed position with all the image edges. After doing that, we know for certain that GROPER has found all image edges that might have come from any of the matched objects that indexing has suggested.

7.2 Computing a Rotation and Translation

To perform verification, the first thing that GROPER must do is to determine what a match tells it about the location of an object. As mentioned in chapter 6, once we have a mapping from at least two non-parallel image edges to two edges in an object, this unambiguously tells us the location of the object in the image. However, due to error, we do not expect that all the matches between image and model edges will indicate exactly the same location. So this section will describe a technique for combining all of the data provided by a match

to determine a rotation and translation of the object model that brings it into a close alignment with the corresponding image edges. The author adapted the code in GROPER that performs this task from code in the system RAF [9] of Grimson and Lozano-Pérez. Their paper contains a more complete description of this process.

This step of verification proceeds in two stages: determining first a rotation and then a translation. We can view the problem as one of finding a three degree of freedom transformation that aligns the two sets of edges, which lie in different coordinate systems. This transformation will consist of a rotation, which gives the two sets of edges the same normal vectors, and then a translation in x and y coordinates, which causes the image edges to lie on top of the model edges.

To find the desired rotation, GROPER simply averages the rotation that would give each model edge the same normal as its corresponding image edge. If two rotation angles differ by too much, then we know that the edges do not have a compatible rotation, and so do not really form a valid match. GROPER makes this decision if two edges produce rotations that differ by more than $\frac{\pi}{2}$. Otherwise, GROPER weights the rotations each pair of edges indicates by the lengths of the image edges, under the theory that longer image edges provide more reliable data about the angle of the edge. GROPER then determines the rotation for this match by taking the normalized average of these weighted angles.

Using this rotation, GROPER then finds the translation that best aligns model edges with image edges. To understand how GROPER does this, we must first understand that a single match between an image edge and model edge provides only one of the two components of translation. To find the translation, we first rotate the model edge about the origin, using the value computed according to the above description. Because the image edge may correspond only to a fragment of the model edge, we may have a range of different translations available that would each align the image edge with a portion of the model edge. However, regardless of which part of the model edge we match to the image edge, the translation will have the same component in the direction of the edges' normals. Differences in the component tangential to the edges will cause the image edge to align with different parts of the model edge. So if we have two matches involving non-parallel image edges, we can find the component of the translation in the direction of two different normals, and solve for it. If we have more than two matches, we can find the translation indicated by every pair of matched edges, and average them. After this, GROPER checks to see that the proposed rotation and translation really does bring each model edge close enough to the corresponding image edges. Grimson and Lozano-Pérez [9] contains a more detailed description of this process for the three dimensional case.

7.3 Finding Additional Support

Once GROPER has calculated the rotation and translation, it can use them to look for extra support for the object. GROPER does this quite simply. It calculates the proposed location of each object model edge in the image, and then compares this to each image edge, to see if the two match. They match if the difference in their angle is less than allowed error bounds, and no part of the image edge lies further away than allowed by error bounds from some part of the model edge. If they match, GROPER adds this pair to its list of matched edges.

7.4 Enough Support?

Having found all the image edges that could match an object model in a particular location, GROPER must decide whether to accept this match. GROPER will accept a match if it accounts for an arbitrarily chosen amount of an object's perimeter, and if it does not conflict with any other acceptable match based on the same initial indexing step. GROPER will reject any match for which the lengths of the image edges do not add up to at least 25% of the perimeter of the object matched. We have chosen 25% because that amount seems to produce good results. If no matches pass through verification, then of course GROPER does not accept any matches, and returns to the grouping step to continue looking for objects. If one match passes this test, GROPER accepts it as valid. If more than one match passes verification, then GROPER compares these matches. If they all use the same image edges, then GROPER decides that it has correctly found all the edges for a single object, but that the image does not provide enough information to tell it which object. In this case, GROPER removes the edges from the image before looking for new objects, and notes from which objects these edges might have come. If several matches account for slightly different sets of image edges, however, then we have no reliable way of preferring one match over the others. Furthermore, accepting an incorrect match would mistakenly remove some edges from future consideration, perhaps making it difficult to find other objects. In this situation, GROPER does not accept any of the matched objects as correct, and instead returns to looking for other objects.

7.5 Problems with Verification

Although section 7.1 argued that GROPER needs a verification step, this step does present some problems. Verification may cause GROPER to incorrectly match some image edges to an object model. This can cause two types of problems. First of all, these incorrect

matches may cause GROPER to falsely recognize an object without really having sufficient evidence. But more seriously, mistakes in verification may remove from consideration edges that GROPER will need to recognize other objects in the scene. We can not expect any recognition system to avoid all mistakes. It is always possible for several objects to produce edges that between them look exactly like some other object. But the verification system discussed in this chapter falls prey to some particularly annoying mistakes that seem avoidable. In particular, this section will provide examples of mistakes that contradict previously discussed ideas of good grouping, and mistakes that do not provide consistent matches. Huttenlocher[13] provides a more formal analysis of some approaches to verification and their likelihood of producing errors.

GROPER attempts to avoid mistakes by using grouping to only consider collections of edges that appear to have all come from a single object. The verification step, however, offers no such guarantee. Verification may find that the modeled object aligns well with some image edge which does not group well with the other image edges matched. In figure 7.1, for example, edge A lies close to the proposed location of a recognized object. However, to a person it seems obvious that edges B and C also came from whatever object produced edge A. And edges B and C do not lie near the modeled object. Such evidence would allow people to avoid adding edge A to the proposed match. GROPER's grouping system also judges that edges A, B, and C probably came from a single object. But we do not have enough confidence in this grouping system to allow it to rule out matches like this. Often GROPER groups together edges which did not come from a single object. These types of grouping mistakes do not cause serious problems when GROPER uses grouping to order a search. But if verification made use of the grouping system, many new errors would result.

Verification also makes errors because image edges may all align with an object model to within error bounds, but still produce global inconsistencies. For example, in figure 7.2, two parallel image edges both lie close to a modeled object, in its proposed site. Either edge would provide a good match to the object, but not both. In general, we would like to avoid matching two sections of image edges to the same section of a model edge. Detecting this mistake can be tricky, however, because an image edge can match a range of different locations on a model edge, depending on how the sensing error has affected the location of the edge. Figure 7.3 provides an example of two image edges which, in one interpretation match the same section of a model, but still offer a consistent interpretation. It also seems difficult to decide how to respond to such a mistake when we detect it. In figure 7.2, for example, we would like a recognition system to accept only one of the two parallel edges as a match, but we have no way to decide which one to accept.

Glaring mistakes also occur when image edges lie close enough to model edges to match them, but do not have the right relationships to each other. Figure 7.4 shows three image

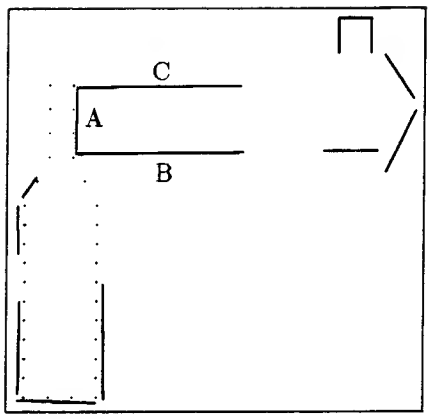

Figure 7.1: Dotted lines indicate the proposed location of an object. A verification step will decide that this object produced edge A, but not edges B and C, even though the three edges appear to have all come from the same object.

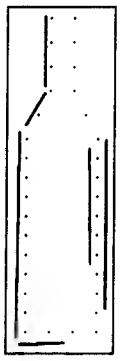

Figure 7.2: Dotted lines indicate the proposed location of an object. The two parallel lines on the right both lie close to it. In such cases, it can be hard to know which is the right match.

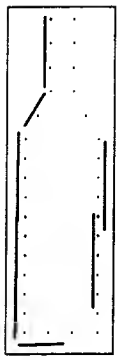

Figure 7.3: In this figure, matching the two right most image edges to the same model edge may create a conflict. But, allowing for error, we can also match them to completely different sections of the edge.

edges which lie close to a modeled object. Error bounds do not prevent GROPER from matching any of these edges to the object model. But they do not look right. Perhaps we should also constrain the image edges to have the same order as the model edges they match, but it can be difficult to determine the order of unconnected image edges. Or we might try to avoid this type of problem by requiring that nearby edges have similar sensing error which brings them into direct alignment with the object model. This would rule out interpretations like the one in figure 7.3. In that example, two image edges can match a model edge to within error bounds, but we would expect that if the two image edges really came from the same object edge, they would look more co-linear in the image. In addition to the problems involved in detecting mistakes due to this kind of global inconsistency, we also have a difficult problem in deciding which matched edges not to use in order to produce a smaller, more consistent match.

7.6 Conclusion

GROPER does not make much of a contribution to this topic. Instead, it makes use of the type of verification performed by existing recognition systems. Because GROPER operates in a domain in which false positive recognitions can cause problems, it does bring to light some difficulties with existing verification techniques. In particular, when we do not assume knowledge of which side of an edge is figure, and which is background, it frequently happens that edges match a correctly located object, even though the object did not really produce those edges. A recognition system which attempts to recognize many objects in the same

Figure 7.4: Each of the three edges at the top matches a different model edge, to within error bounds. But something looks wrong.

scene will encounter difficulties when mistakes like this cause it to remove from future consideration edges that it will later need.

Chapter 8

Previous Work

8.1 Introduction

This chapter discusses how previous work in computer vision relates to GROPER. It has two primary functions. First of all, it will acknowledge work that inspired many of the ideas in this thesis, while making clear what aspects of GROPER are novel. And secondly, the experience gained with GROPER can help us to understand why previous object recognition systems have worked well in their domains.

GROPER differs from past recognition systems primarily in the way it performs grouping, and in the way it uses grouping. In retrospect, we can see that many recognition systems use grouping to eliminate from consideration all but a small class of combinations of edges in an image. GROPER differs from these approaches by using grouping to impose an order on the set of edge groups, instead of just using grouping to make a yes or no decision about whether to consider a collection of edges. This explains why GROPER needed a mechanism for estimating the probability that any collection of edges originated in the perimeter of a single object. Because they mostly used quite simple grouping methods, and because they did not want to risk missing too many objects, recognition systems that have used grouping to make a yes or no decision about what groups of edges to consider have said "yes" to a lot of groups. As we have seen before, an unordered search through a large number of sets of edges leads to a combinatorial explosion and many false positive identifications when dealing with complex recognition tasks.

This chapter will discuss four types of previous work. First of all, many scientists have worked on the problem of segmenting images into relatively homogeneous component parts. Section 8.2 will briefly discuss that work. Although not directly relevant to the approach taken by GROPER, this work may have relevance to possible extensions to GROPER. Section 8.3 will discuss past work on the perceptual organization of edges. This work

has provided important motivation for the grouping system developed in GROPER, both by pointing to important grouping phenomena, and by suggesting ways to explain these phenomena. Thirdly, we will discuss some previous approaches to the problem of object recognition. Only Lowe has explicitly made use of grouping in a recognition system, and his work has provided inspiration for GROPER. But we will also see that we can view many other recognition systems as containing implicit grouping components. Finally, we will examine other approaches taken to the problem of indexing into a library of objects, or otherwise making use of such a library. Most indexing systems either assume an already segmented image, or can handle only special collections of edges, such as connected edges, or unoccluded edges. GROPER's indexing system differs from others mainly in that it can deal with arbitrary collections of edges. This section will also discuss why the use of indexing makes grouping particularly important. Throughout the chapter we will stress that other work on grouping has provided important motivation for GROPER, but that GROPER uses grouping in quite a different way from these other systems.

8.2 Segmentation

Much work in computer vision has focused on the problem of segmenting images. The goals of that work, although superficially related to this thesis, actually differ from it quite a bit. Most work in image segmentation seeks to break an image up into homogeneous parts. It does not help to tell whether two parts, separated in the image, actually come from a single, unoccluded object. Nor does it help us when several different materials make up the surface of a single object. Traditional segmentation attempts to divide an image based on the materials in it, rather than the objects.

Haralick and Shapiro[12] survey some segmentation systems that attempt to divide images into sections of relatively homogeneous intensity levels. For example, as a simple approach one can find clusters in an intensity histogram of an image, and then form segments from the connected pixels whose intensities all fall within the same cluster. See Carlotto[6] for a recent approach along these lines. Alternately, the split and merge approach breaks apart segments that are not sufficiently homogeneous, and combines similar segments (see, for example, Pavlidis[21]). Researchers have developed many other approaches as well. For our purposes, the point is that these approaches to segmentation only produce segments that do not have too great a variation in their intensity.

Three reasons make these types of approaches inapplicable to GROPER without significant modification. First of all, they do not directly provide the information that GROPER needs, the likelihood that two unconnected segments in the scene originated in the same object. Secondly, in principle, these methods cannot handle objects made from more than one

material. A good segmentation system wants to separate the wooden handle of a hammer from its metal head, and will not tell us anything about the likelihood of the two segments they form coming from the same object. And thirdly, work on segmentation has focused on determining a single, optimal segmentation of the scene. Many applications require such an output. But GROPER's approach to recognition assumes that any single decomposition of a scene will probably contain mistakes. So GROPER requires an ordering of the space of all possible segmentations. After getting nowhere with the best segmentation of a scene, GROPER wants to know the second best segmentation, so it may try that next. For this purpose we need an approach to segmentation that can assign a probability to any segmentation, or partial segmentation of an image.

However, a more sophisticated grouping system might make use of some previous work on segmentation. First of all, we might use segmentation to derive good primitives for grouping. A conservative segmentation algorithm might produce, with high reliability, small sections of the image extremely likely to come from the same object. Such a system would only fail when two adjacent or occluding objects have similar intensity levels, which may not happen often. These primitives might prove as reliable as the nearly connected convex edge sections used by GROPER, and more informative, because they allow the extraction of intensity and texture information. Secondly, segmentation work has developed machinery for determining whether the intensities of two regions of an image indicate that they come from the same object. The split and merge approach, for example, must make a judgment about whether two regions of an image are similar enough to justify merging them. One might be able to modify this work to provide a probability, instead of a strict yes or no answer.

Although they do not answer the questions GROPER needs to ask, segmentation systems do provide much insight into the use of intensity information in grouping. As mentioned in chapter 3, possible extensions to GROPER would find such insights quite valuable.

8.3 Perceptual Organization

Much work by psychologists and computer scientists has investigated human grouping phenomena. This work has contributed to the ideas in this thesis primarily by suggesting explanations for grouping phenomena, and by identifying various grouping phenomena.

Two explanations of grouping have particular relevance to computer scientists attempting to use grouping phenomena in a recognition system. First of all, it has been suggested that people tend to group together parts of an image particularly likely to have come from a single object. This thesis focuses on that type of grouping phenomena. It has also been suggested that people form groups that will prove particularly useful to the remainder of

the recognition process. In this view, one does not arbitrarily group any edges coming from the same object, rather one groups to identify a useful relationship between some of the edges in an object. These views do not contradict each other. One may explain some grouping phenomena in one way, and other phenomena in the other way, or consider that both factors contribute to certain grouping tendencies.

Lowe[18] shows that grouping can form collections of edges particularly useful to the recognition of three dimensional objects. For example, grouping together parallel edges proves useful because, under certain viewing assumptions, parallel edges in an object appear parallel in any image of that object, regardless of its orientation. Once one has identified two parallel edges in an image, one need consider matching them only with parallel edges in the object. So one can also explain grouping as a way of creating good primitives of comparison for recognition. Biederman[1] also suggests using this type of grouping in a recognition system.

The Gestalt psychologists, who first studied grouping, suggested that one groups together parts of an image that tend to correspond to objects in a scene. Kohler in *Gestalt Psychology*[16], his review of the work of the Gestalt movement, points out that we group together adjacent surfaces with common surface properties, and that such surfaces have a greater likelihood of coming from a single object than do adjacent surfaces with different properties. Witkin and Tenenbaum[27] discuss the hypothesis that we group together things that have a relationship unlikely to occur by accident. These grouped image sections, they argue, probably come either from a single object or a single process, such as the symmetric streaks caused by a single windshield wiper. And Lowe [18] has attempted to demonstrate mathematically that parallel edges in an image have a greater likelihood of coming from a single object than do non-parallel edges.

This thesis does provide somewhat more explicit reasons for suspecting that certain collections of edges come from a single object. Instead of just asserting that nearby edges in an image are likely to come from a single object, we attempt to provide an analysis of the image formation process to back up this hypothesis. This approach also provides a more quantitative analysis of grouping.

The Gestalt psychologists pointed out a number of different grouping phenomena, including the distance constraint used in GROPER. However, past work has not taken notice of the orientation constraint, as we develop it here. Although it does not produce as striking results as constraints discussed by the Gestaltists, informal psychophysics do support the hypothesis that people make use of a similar constraint in grouping. This constraint was motivated by a parts based view of objects, which others have investigated. Many researchers have suggested that we can simply represent objects as composed of a small number of convex parts. Whitman Richards particularly emphasized the relevance of this

view to grouping in personal communications.

Finally, Lowe[18] has emphasized the need for grouping to reduce the computation needed for object recognition. We also emphasize its value in reducing false positive identifications of objects.

8.4 Recognition

Many computer systems have approached the problem of visual object recognition as a search through the space of possible mappings from image features to object model features. More recently, some researchers have recognized the importance of trying to perform grouping in a scene before referring to object models. However, a new look at even the more traditional recognition systems reveals that many of them contain an implicit grouping process to control the complexity of the search for objects. Besl and Jain[3] and Binford[2] provide good summaries of work on model-based object recognition. This section will review some of those systems with an eye towards understanding how grouping helped them work, and then describe how a more explicit grouping step has benefitted some recognition systems.

ACRONYM [5] was one of the earlier model-based object recognition systems. It modeled objects as generalized cylinders, and then attempted to match model cylinders to possible projections of such structures in the image. For example, one might model an airplane as having a cylindrical cabin. A cylinder can project in an image as either an ellipse, if seen from one end, or as a rectangular "ribbon" if seen from the side. ACRONYM first located possible ellipses, or ribbons in the image. It then considered the possible matches between these image features and the generalized cylinders of the object. This describes only a small part of ACRONYM's capabilities, but we will focus on that part.

ACRONYM performed grouping by looking for collections of edges that might correspond to a part of the object for which it searched. It might combine several edges into a ribbon, for example. This greatly reduced the amount of search ACRONYM needed to perform. Instead of a large number of edges in the image and the model, ACRONYM dealt only with a much smaller number of primitives, each of which contained several edges. Although effective in its domain, this type of grouping has two problems when applied to a more general domain. First of all, it depends on selecting a few primitives, such as generalized cylinders, as the parts of the objects for which the system looks. If one tries to recognize objects not made of such parts, ACRONYM's method of grouping will not form collections of edges that all come from these objects. And secondly, ACRONYM's grouping system appears to depend on an entire generalized cylinder being visible in the image. Brooks demonstrated ACRONYM working mainly on unoccluded objects, such as

airplanes on the ground viewed from above. If something occludes a part of an object, the grouping component may not form a primitive corresponding to that part.

Bolles and Cain[4] use a distance constraint to reduce the amount of search needed by their Local-Feature-Focus method (LFF). After hypothesizing a match between an image feature and a model feature, LFF looks for other features in the image near this initial one. It then determines which model features these image features might match, and finds the largest mutually consistent subset of these collections of matches. LFF uses grouping by only considering a collection of features near each other as having potentially all come from a single object. This approach reduces the computation needed for recognition by greatly limiting the sets of matches between image and model features that LFF needs to consider. This type of grouping is also effective because, as we have seen, nearby features have a greater likelihood of coming from a single object than more distant features, so LFF focuses on the more likely hypotheses. However, LFF suffers from the disadvantages of any all or nothing grouping scheme, that is, a scheme that only decides it will or will not consider a hypothesis, instead of ordering the hypotheses. If LFF focuses only on sets of features quite close to each other, it runs the risk that an object will not produce enough nearby features in the image to make recognition possible. On the other hand, if LFF considers larger collections of features, the number of combinations of matches it must consider will grow. Since LFF does not order these combinations, it must consider them all before making a decision. Furthermore, Bolles and Cain realize that they can not expect a collection of nearby features to all come from the same object. So instead of looking for a match that accounts for all the features in a local cluster, they must look for a match that accounts for the largest subset of these features, a more computationally intensive task. GROPER avoids this problem by using additional grouping clues to form collections of edges hypothesized to all come from a single object.

RAF, by Grimson and Lozano-Pérez[10], performs grouping specific to the object it looks for, using a Hough transform. To avoid considering all combinations of matches between image and model edges, RAF begins with a clustering phase. It divides the space of all rotations and translations into buckets. Then, for each match, RAF determines the range of rotations and translations that would align the model edge with the image edge, and places a pointer to the match in all the buckets corresponding to potential rotations and translations. After doing this for all matches, RAF searches for an object by first trying the buckets that contain the most matches. Each bucket should contain only roughly compatible matches, and the search then finds the largest set of matches that passes a stricter test for compatibility. Although effective in its domain, this type of grouping does not work well when one has knowledge of a library of different objects. If one performs clustering for each object model, the clustering will take a lot of time, and may well produce a large number

of collections of matches to consider. This approach also does not help reduce false positive recognitions too much, because it does not prevent RAF from considering collections of edges not really likely to have come from a single object. For example, GROPER would not consider an interpretation that used only one edge in a square of four edges, because the four edges would form a primitive. RAF's grouping technique does not prevent it from considering such possibilities.

Lowe's system, SCERPO, differs from the above systems in that it explicitly makes use of grouping. It selects small groups of edges that have a good chance of coming from a single object, and that provide good recognition primitives, as we previously described. Lowe relies on grouping to limit his search more than some of the previously mentioned systems, because grouping not only limits the number of collections of image edges his system SCERPO must consider, but it also allows SCERPO to only consider matching each group to a small number of sets of object edges. It does this because it knows that, for example, it need only match parallel image edges to parallel object edges, or three co-terminating image edges to three co-terminating object edges. This reduction in search allows SCERPO to handle the more complex domain of recognizing three dimensional objects in two dimensional images. However, after selecting a limited number of initial groups, SCERPO then considers all viable matches in an unordered search. So, SCERPO must perform an appreciable amount of search.

In summary, ACRONYM, LFF, SCERPO and RAF all make use of grouping to limit the amount of search needed to recognize objects. And these methods work well in their domains. However, they all have the difficulty that, while they restrict the needed search to manageable proportions, they still require an undirected search through a relatively large space of possibilities. These approaches still leave open the problems of growing computational complexity when used with libraries of objects, and do not fully address the problem of false positive identifications of objects.

Instead of using grouping to just limit search, some recognition systems attempt to use segmentation to completely separate the image of an object from its background. This approach obviously makes recognition much easier. It does not require a search through a set of possible segments in the image, and will not produce false positive identifications caused by attempting a match involving two sections of the image unlikely to have come from a single object. This kind of segmentation has only been done in limited domains. In those domains, however, segmentation can provide an effective first step in a recognition system.

Segmentation can work well when one has access to three-dimensional image data. Yang and Kak[28], for example, present a system that separates the topmost, convex object in a pile, using three dimensional data acquired with a laser scanner. This presents a somewhat

simpler problem then segmenting two dimensional images, however, because one may look for discontinuities in depth or curvature. Two dimensional images do not make this type of information available.

Gross and Rosenfeld[11] are working on the problem of segmenting objects in two dimensional scenes for the purposes of recognition. However, they restrict their domain to one containing unoccluded objects of uniform intensity.

Segmentation of general two dimensional images presents quite a difficult problem. For this reason, GROPER takes a search-based approach, which attempts to explore the most likely segmentations first, but does not depend on determining a single, correct segmentation.

Previous recognition systems have not noted the problem of false positive recognitions, which we have found to present a serious challenge. The use of some grouping in these recognition systems may partly explain the fact that they have not produced significant numbers of false positive identifications. However, we will now argue that some special characteristics of the domains of many object recognition systems have allowed them to avoid this problem.

Three aspects of GROPER's domain make false positive identifications a potentially significant problem. First of all, GROPER looks for any object in a library, and not just a single object. Secondly, GROPER does not assume that it can tell, before recognizing an object, which side of an edge is part of that object, and which side belongs to the background. And finally, GROPER can deal with simple objects, and with scenes containing objects similar in appearance to the ones for which it looks. Each of these three factors make false positive identifications much more common in GROPER's domain than in that of many other recognition systems.

Obviously, the more objects one knows about, the greater the chances of mistaking some randomly chosen edges in the scene for one of those objects. We just wish to note that GROPER works with relatively simple libraries; false positive identifications will become a much greater problem with libraries of thousands of objects that have flexible or parameterized parts.

Not knowing figure from background greatly increases the number of combinations of edges to consider because each edge has two possible interpretations. This increases the chances that some such combination will look like an object in the library.

Finally, the objects for which a recognition system looks, and which it uses in scenes, can make false identifications particularly unlikely. A collection of edges from different objects have little chance of looking like a complex object, which has many edges. They have a much better chance of resembling a simple object, with few edges, like the ones used in GROPER's object library. Furthermore, we have tested GROPER on images that

contain objects similar to some of the objects in the library. For example, if a recognition system knows about two objects with common subparts, it takes less of a coincidence for one of these objects to look like the other. Many recognition systems are tested in a domain in which they look for a single complex object, in a scene containing that object and a few very different looking objects.

8.5 Indexing, and Libraries of Objects

This section will discuss four other recognition systems that have performed indexing. GROPER's indexing system makes use of some ideas already explored in these systems. However, GROPER's indexing differs from these systems in ways that make it fit well with the type of grouping that GROPER uses. This section will also briefly discuss some other approaches to recognition that handle libraries of objects.

Kalvin et. al.[14], Wallace[25], and Thompson and Mundy[23] use indexing in recognition systems. They all describe a collection of edges in an image with a small number of parameters that do not depend on the orientation of the edges. They then look in a table to find object models that could have produced those image edges. However, the way these systems work prevents us from applying them directly to the kinds of groups of edges produced by GROPER. The system of Kalvin et. al. identifies objects that might have produced a single connected section of contour. It does not provide a way of using the information contained in the spatial relationship between two sections of contour presumed to come from a single object, or make provisions for objects that do not contain distinctive sections of contour. Wallace does make use of this type of information. Although unknown to the author during the development of GROPER, Wallace's approach is quite similar to GROPER's, but it only handles indexing that uses two unoccluded sections of an object's perimeter. GROPER makes use of occluded sections of perimeter, and can combine the results of many separate indexing steps to find a globally consistent interpretation for a group of edges. Thompson and Mundy have developed an indexing scheme that can use only pairs of vertices in an image. They need not handle occlusion, since they assume they will find unoccluded vertices. GROPER makes use of more general collections of edges. However, Thompson and Mundy designed their indexing system to recognize three dimensional objects.

Kalvin et. al. have an indexing scheme particularly suited to identifying distinctive unoccluded sections of an object's perimeter. They divide the curved contour of an object model into discrete sections. They then describe each section of contour using five parameters that do not depend on the orientation of the object. These parameters describe the curvature of a short section of contour. Then for each such short section of perimeter in

each object model, they make an entry in a virtual five dimensional table which they actually implement as a hash table. To identify which contours from known objects might have produced a section of image contour, they perform a table lookup for each short section of the image contour, and combine the results. They could just take the intersection of these table lookups to find a section of model contour that might have produced the entire image contour, but in fact they use a more complicated heuristic to handle noisy data. This system has two important advantages over the indexing GROPER performs. First of all, it can handle curved data directly, instead of first approximating it with straight lines. Straight line approximations can introduce many types of error in the recognition process. And secondly, its library only needs to contain a number of entries dependent on the length of an object's perimeter. GROPER stores a number of entries proportional to the square of the number of straight line approximations to the perimeter. The chief disadvantage of Kalvin et. al.'s approach is that it does not use the relationship between disconnected image segments thought to come from a single object. GROPER's grouping system focuses on producing just this kind of information. This limitation suits Kalvin et. al.'s approach to the recognition of objects, which expects to find connected sections of contour visible that will distinguish objects from most of the others in the library.

Wallace combines indexing with a clustering technique. Wallace's system uses indexing to allow each pair of contour segments in an image to cast a vote for each object that might have produced it. The system then searches first for the object that gets the most votes. Indexing also produces matches between image contour segments and model segments that can guide the resulting search. Wallace indexes using both straight line approximations to contours in the image and also curved arcs that appear in contours. We will consider only the indexing that uses straight line approximations, because it most closely resembles GROPER.

Wallace indexes with straight lines in the image in much the same way that GROPER does, except Wallace assumes that these lines will correspond to unoccluded sections of object perimeter. So, Wallace represents two non-parallel straight lines with three parameters: the angle they form, and the distance from the mid-point of each line to their intersection point. An occluded edge will have a different mid-point than the same, unoccluded edge. This does not present a problem to Wallace's system, which only plans on using pairs of unoccluded edges from the same object to assist in recognition. Wallace then builds a three dimensional table, containing an entry for every pair of edges in each object model. An indexing step can then tell the system which unoccluded pair of model edges could have produced a pair of edges in the image. This characterization does not capture all the information in a pair of edges, because two edges of different lengths might have the same mid-point. Wallace only puts a single entry in the table for each pair of edges, resulting in

a sparse table. However, this means that Wallace's system does not explicitly account for error.

GROPER's indexing scheme primarily differs from Wallace's in that it handles occluded edges and accounts for error. Instead of making entries in a table for the way two unoccluded edges may look, GROPER makes entries for the way they may look given every possible occlusion of the edges and given all possible sensing error. This results in a more dense table, but allows GROPER to use occluded edges in its recognition process. Also, since Wallace uses only unoccluded edges, the need to check the results of indexing for global consistency does not arise. GROPER uses indexing to find all the globally consistent matches to a group of edges. Chapter 6 describes how GROPER does this.

Thompson and Mundy[23] also build a table that shows all possible relations between a pair of object features. Given two vertices in a three dimensional object, they use six parameters to describe the relationships these vertices can have when projected into a two dimensional image. They then determine the values these parameters have when the object is looked at from all view points, sampling the space of all view points at five degree intervals. They then make a table entry for every view point. This results in a great many table entries, and may cause some inaccuracy if the appearance of a pair of vertices changes much between two view points. This type of approach does not apply to GROPER's domain, since GROPER needs to handle only two dimensional objects, but must index using arbitrary collections of edges.

Ettinger[7] takes an approach somewhat similar to Wallace's, based on object parts. Ettinger's system decomposes each object in the library into sub-parts, and first looks for these sub-parts in the image. One sub-part may appear in many objects. The system then detects features in the image, such as corners or bumps, at a coarse scale. Each feature votes for the existence of all sub-parts that have that feature. The system then tries to recognize each of these sub-parts, starting with the ones that have received the most votes. Like the approach of Kalvin, et. al. Ettinger's approach indexes using only sections of unoccluded contour from an object. Ettinger's system also has some interesting properties that allow it to recognize classes of similar objects, to a limited extent, but this aspect of it falls outside the domain of this discussion.

We will discuss two other systems that have approached the problem of using libraries of objects in recognition without using indexing. Turney et. al.[24] and Knoll and Jain [15] have used sub-template matching to locate contour sections in an image that match some contour sections of objects in the library. They then use these matches as a basis for finding objects in the image. We will particularly discuss the relevance of these approaches to the problems of accuracy and computational complexity. We previously argued that the use of object libraries will make these problems particularly serious.

Turney et. al. have described a recognition system that performs clustering on the results of sub-template matching. For each object in a library of objects, this system selects some sections of the object's contour that particularly distinguish that object from others. Turney et. al. refer to these contour sections as sub-templates. Their system looks in the image to find all the sub-templates that appear as part of image contours. For each sub-template found, the system can determine the rotation and translation that would align any object model that has a similar contour section with the contour section in the image. So the system uses each sub-template match as a vote for a particular object's presence at a particular orientation in the scene. If a sub-template matches only a few object models, it casts a big vote for each of those objects. If it matches many models, it casts a weaker vote for those objects. The system selects the model and orientation that gets the most votes.

The amount of work required by this system will grow with the size of the library of objects. Presumably, the more objects one knows about, the more sub-templates one must look for in the image, if one wishes each sub-template to be salient, that is, to look like relatively few sections of object models. Turney et. al. seem to rely on using salient sub-templates to avoid having too many contour segments voting for objects that did not really produce them. The rate of growth, however, will depend on the particular strategy used for choosing sub-templates.

For accuracy, Turney et. al. seem to rely on finding a significant number of image contour segments that could not have come from too many different objects in the object library. Their system can not combine information contained in unconnected segments in the image. One can imagine many real images in which combining this kind of information would greatly improve the accuracy of a recognition system. GROPER, for example, works using similar objects with very simple contours. A relatively long segment of one of these objects may consist of a single straight edge, or of two straight edges joined with a right angle. This type of domain can cause a problem for Turney et. al. because if a sub-template matches such a simple segment, it will have to vote for most of the objects it knows about, and if it does not, it will not use that segment to help it recognize objects.

Knoll and Jain[15] present a system that uses sub-template matching to generate a large number of hypotheses. They then perform a verification step on each hypothesis to locate the good ones. So they select sub-templates that match a large number of different objects. In fact, they want sub-templates to match $O(\sqrt{N})$ different objects, where N indicates the number of different objects that the library knows about. They select $O(\sqrt{N})$ such sub-templates, so that each object can produce some sub-templates. They look for these sub-templates in the image. Each one they find provides them with a hypothesis for the location of $O(\sqrt{N})$ objects, and they attempt to verify each hypothesis. They then accept

the hypotheses that receive the most verification.

This algorithm allows Knoll and Jain to deal elegantly with the computational complexity of the recognition process. Both the process of locating features, and the process of verifying hypotheses will take only $O(\sqrt{N})$ time. Furthermore, since they can look for each feature independently, with a parallel architecture they could assign a processor to the job of looking for just one feature. With $O(\sqrt{N})$ processors, they could locate all the relevant features in time dependent only on the image size and the model size. Similarly, with that many processors they could perform all the verifications in time that depends only on the image size. Thus, their approach solves much of the combinatorial problems of recognition, particularly when combined with parallelism.

On the other hand, although parallelism allows us to run Knoll and Jain's algorithm in constant time, we might still want to determine the minimum amount of computation required to perform recognition, independent of the number of processors required. So their system does not completely solve the problem of performing computationally efficient recognition.

Also, their system may encounter two problems relating to accuracy when applied to larger libraries of objects. First of all, the larger the library used, the smaller the percentage of objects in the library a sub-template should match. To find sub-templates that match a smaller percentage of objects in the library, the sub-templates will become larger. But as they become larger, the sub-templates will have less chance of appearing in the image unoccluded. And to recognize an object, the object must produce an unoccluded section of contour corresponding to a sub-template. This problem will become more acute if a library contains flexible objects, which can match a wider variety of small sub-templates. This type of problem will depend a good deal on the particular nature of the object library used, and so it is hard to predict its severity. But in general, it seems that large libraries of objects will present a problem to recognition systems that rely on finding reasonably distinctive, connected contours in the image.

Secondly, Knoll and Jain's approach does not address the problem of accuracy. Their approach relies on generating a large number of hypotheses, a number that grows with the size of the object library. But, as we have attempted to show, this will result in increasing numbers of false positive identifications. The more hypotheses we consider, the more we will find that coincidentally receive support from different objects in the scene.

8.6 Conclusions

This chapter has examined some other recognition systems from GROPER's point of view. It points out that many successful recognition systems have used some type of grouping to

improve their performance. These systems provide additional evidence that some form of grouping can assist in the recognition process. We also argue that no one has presented a system that solves all the problems inherent in performing recognition using libraries of objects. Several systems present interesting ways of coping with the computational complexity of recognition, particularly the indexing approaches of Kalvin et. al. Wallace and of Ettinger, and the approach of Knoll and Jain. However these systems may not perform robustly on scenes containing objects with significant occlusion using large libraries. And these approaches do not address the problem of false positive identifications that large libraries of objects create.

Chapter 9

Results and Conclusions

9.1 Introduction

This chapter will present the results of some experiments with GROPER, and draw some general conclusions regarding its performance. Empirical results will measure the speed and accuracy of three different aspects of GROPER, as well as the amount of storage GROPER requires. We will then attempt to characterize the kinds of problems on which we can expect a grouping system like GROPER's to work well. Finally, we will make another attempt to argue that for complex recognition tasks we will need some type of effective grouping system, even if these tasks prove too difficult for GROPER's grouping system.

First of all, we will look at how well GROPER's grouping system works in various types of situations. We test the grouping system on scenes containing untextured two dimensional objects with fairly uniform lighting. In such scenes, most intensity edges arise from occluding contours. We also test the grouping system on scenes with three dimensional objects. These scenes also produce intensity edges resulting mainly from occlusions. Finally, we examine GROPER's performance on more realistic three dimensional scenes that produce many different kinds of intensity edges.

Secondly, we will measure GROPER's indexing capabilities. We want to know how much space the indexing tables typically require. Also, we want to know about its accuracy and speed. GROPER's indexing system ensures that, except for bugs, indexing will always produce all sets of model edges that might have produced a group of image edges. So, in terms of accuracy, we wish to know how often indexing will produce a set of model edges that will not align with the image edges for any rotation or translation of them. And, since indexing requires GROPER to perform several intersections, we wish to know the expected cost of these intersections as well as the possible benefits of other implementations.

Thirdly, we will examine how much GROPER's grouping system benefits it by looking

at its accuracy at recognition and the amount of computation it needs. We do this by comparing GROPER's performance to that of an alternate recognition system, SEARCHER. SEARCHER differs from GROPER only in that it uses a backtracking search instead of grouping to decide which collections of image edges to use in finding objects.

Finally, we will conclude by considering the use of grouping in two different domains. We will argue that GROPER's grouping system can greatly benefit a recognition system that operates in a domain where we have a method of locating the occluding edges in an image, and where we have somewhat limited occlusions. Then we will argue that, although GROPER's grouping system will not perform well in more complex domains, it does provide a basis for implementing more elaborate grouping systems. And, we will maintain that some more elaborate grouping system will prove essential in building recognition systems with truly human performance.

9.2 Results of Grouping

9.2.1 Description of Tests

GROPER uses grouping to order its search for objects. We can judge the grouping system by how quickly GROPER succeeds in recognizing objects, but we can also evaluate the grouping system on its own, by seeing how frequently the first groups that GROPER decides to investigate really all come from the same object. This will provide us with more precise information, and also allow us to try out GROPER's grouping system on scenes that contain three-dimensional objects, which GROPER cannot recognize.

In particular, we have examined the pairs of convex sections of edges that GROPER decides to combine together. Other grouping issues affect GROPER's performance, since it often recognizes objects using groups that contain only one convex section, or more than two sections. But the rate at which GROPER can find correctly interpreted pairs of convex sections plays the major role in GROPER's overall performance. Also, looking at just this problem, we can see how well the distance and orientation constraints work.

We have tested the grouping system by looking at the thirty groups that GROPER found most likely to have come from a single object. We decide GROPER was right in choosing a group if all the edges in the group really did come from a single object. Furthermore, when GROPER forms a group, it makes a decision about which side of each edge the object lies on, and which side is background. So we only decide that GROPER has made a correct decision if it has also made a correct figure/background judgment for each edge.

We ran these tests on six sets of images, of varying degrees of difficulty, which we will refer to as test sets one through six. Four tests involved five images each, the others used

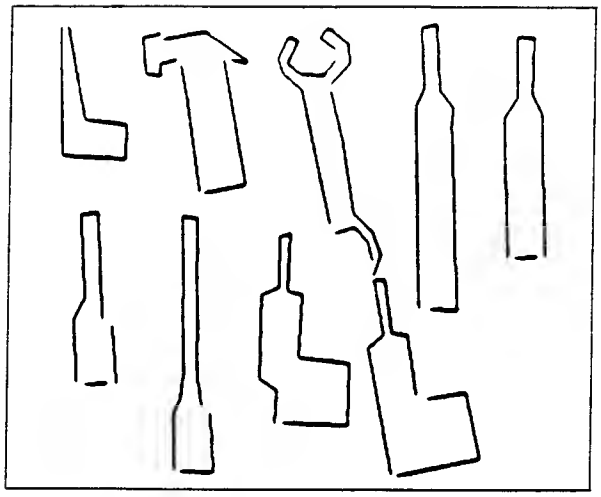

Figure 9.1: The perimeters of the models used in the first set of tests of GROPER.

two and three images. We tried GROPER on four sets of images made from polygonal, two-dimensional objects. Shapes cut out of black paper appeared on a white background in these images. GROPER worked best on these images because most of the edges in them came from the perimeter of objects, although a few edges would appear due to shadows or slight variations in the background. Figure 9.4 shows an example of one of these scenes, and figure 9.5 shows straight line approximations to the intensity edges in it. We made these tests using different assortments of objects. In test one we used the nine objects whose edges are shown in figure 9.1, and a couple of additional objects, unknown to GROPER. For test two we used scenes containing eight of the sixteen objects shown in figure 9.2. A third set of images contained all sixteen of these objects. And a fourth set used the six objects shown in figure 9.3.

Next, in test five, we tried GROPER on scenes made up of real objects. These objects also had uniform reflectance, producing no texture edges. And they appeared on a white background. But they were three dimensional, and curved. Because of their three dimensionality, lighting effects created many edges that did not come from the perimeter of any object, including edges from shadows, specularities, and changes in surface orientation. Figures 9.6 and 9.7 show examples of one such image, and the edges it produces. As we will see, GROPER performed less well on these images, probably due to the more complex edges they produced.

Finally, for test six, we tried GROPER on two images of real-world scenes, containing all sorts of textured objects, and varying lighting effects. GROPER did least well on these images, and, in fact, even a person may find it difficult to interpret the straight line approximations to edges that result from this type of image. Figures 9.8 and 9.9 provide

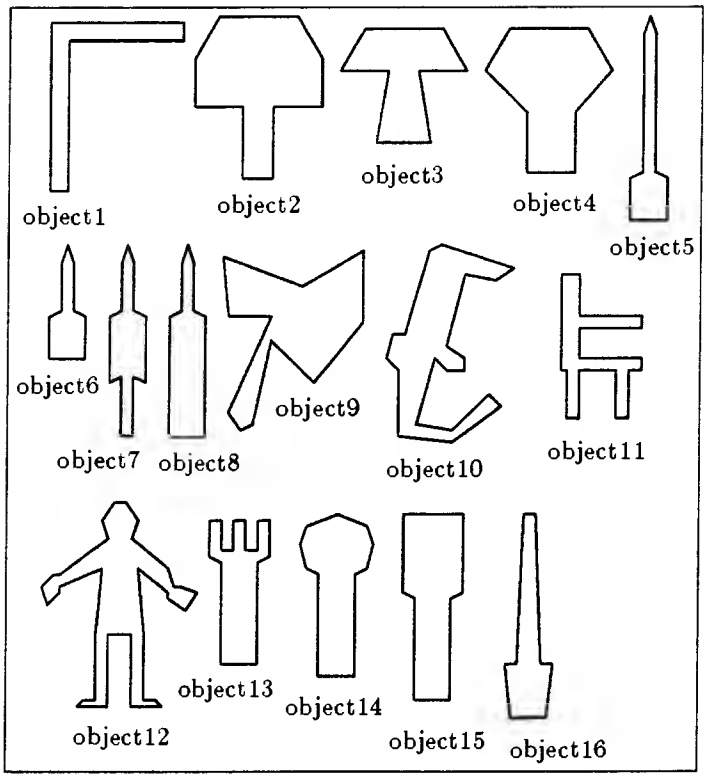

Figure 9.2: The perimeters of the models used in the second and third sets of tests of GROPER.

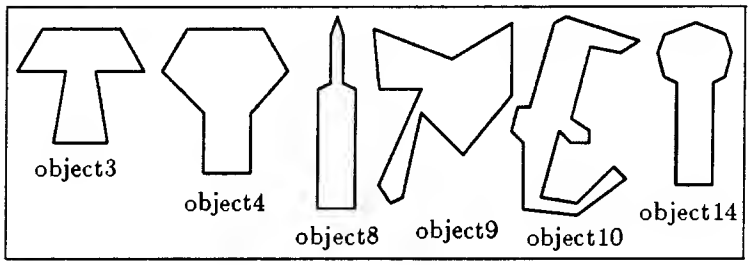

Figure 9.3: The perimeters of the models used in the fourth set of tests of GROPER.

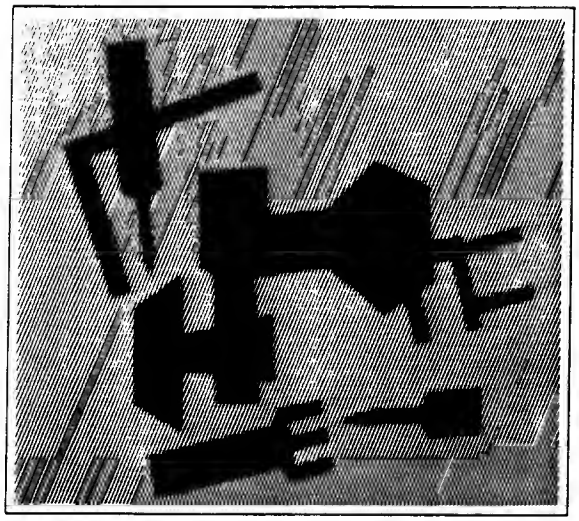

Figure 9.4: A scene from the second set of tests. It contains the objects in figure 9.2.

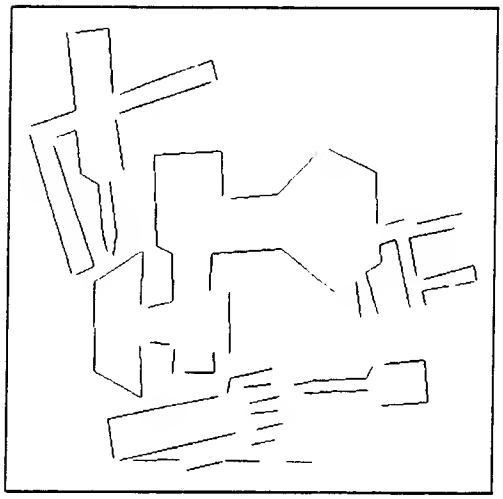

Figure 9.5: Straight line approximations to the intensity edges found in the image in figure 9.4.

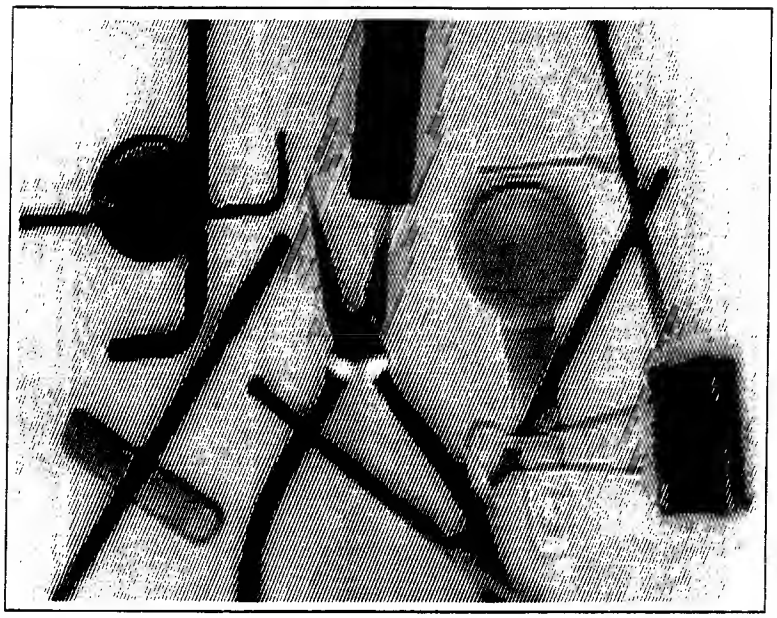

Figure 9.6: An image of three dimensional objects on a white background. The white outline indicates the edges found in the image.

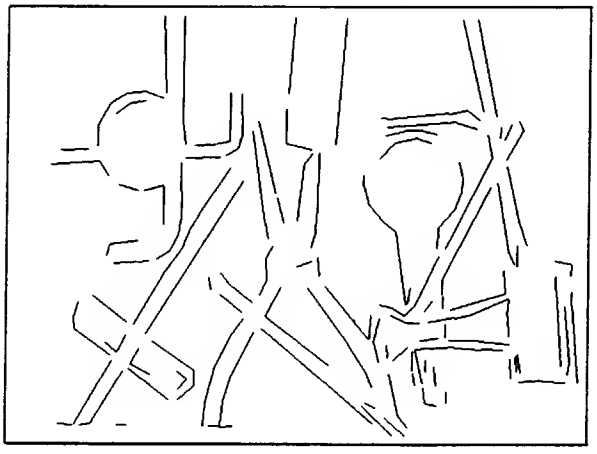

Figure 9.7: Straight line approximations to these edges.

Figure 9.8: An image of a room, containing an arbitrary set of objects.

an example of this type of image, and the edges it produces.

9.2.2 Discussion of Results

Figure 9.10 summarizes the results for these six different sets of images. These results tell us that GROPER's grouping system works quite well on scenes containing two dimensional objects, particularly in comparison to the random selection of groups of edges. There are also some problems with the way GROPER's grouping performs on these images, which these tests do not reveal. We will discuss those problems in section 9.4. The grouping tests do tell us that GROPER works somewhat less well on scenes with more objects, such as the ones in set three (see figure 9.11 for an example), and on scenes with three dimensional objects. Still, we get good results on these images, and a random strategy would also do more poorly on these images than on those of the other two dimensional test groups. Finally, we see that GROPER does not perform well on the more realistic three dimensional scenes of test six. This seems to be due to the poor quality of edges found in the scene; these edges do not seem to provide enough information to even allow people to easily recognize objects.

For tests one, two and four, GROPER averages between 7.8 and 8.8 correct pairs of convex sections chosen out of thirty. To evaluate these results, we need to get a rough idea

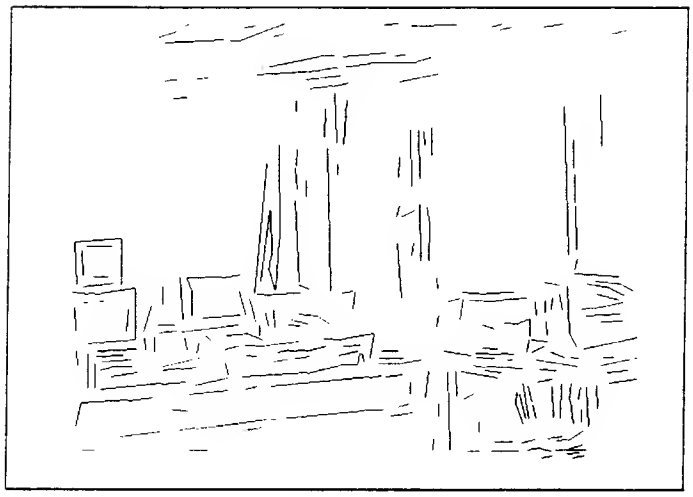

Figure 9.9: Straight line approximations to the edges in this image.

	2D Objects, Test 1	2D Objects, Test 2	2D Objects, Test 3	2D Objects, Test 4	3D Objects,	Real Scenes
Number of Images	5	5	3	5	5	2
Of Thirty Groups Preferred, Average Number That Were Correct	8.8	8.4	5.67	7.8	5.8	.5

Figure 9.10: The results of grouping experiments.

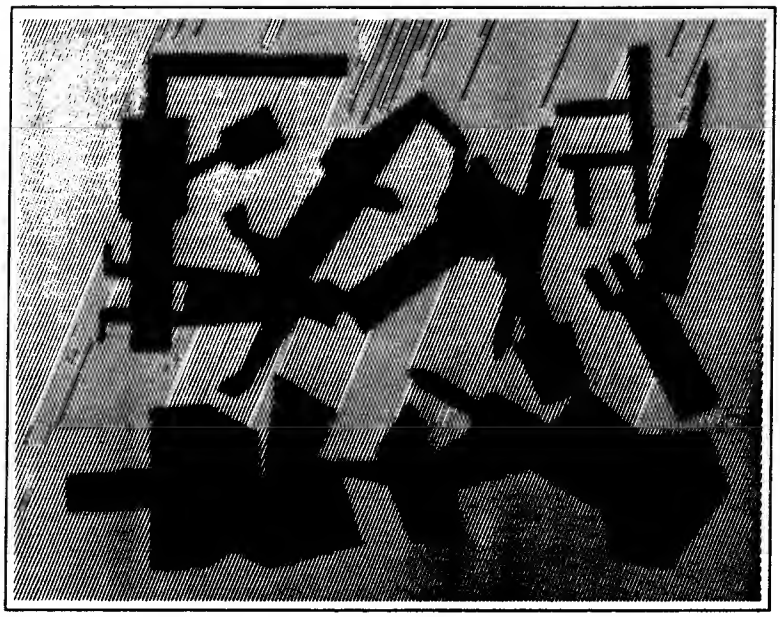

Figure 9.11: A harder two dimensional scene, with sixteen objects

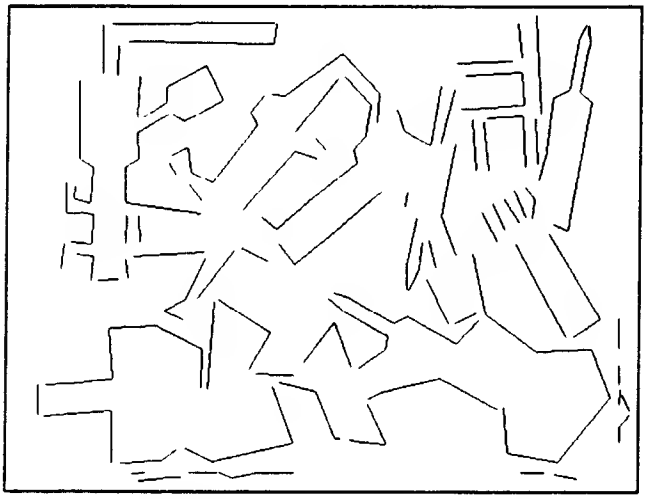

Figure 9.12: The edges it produces.

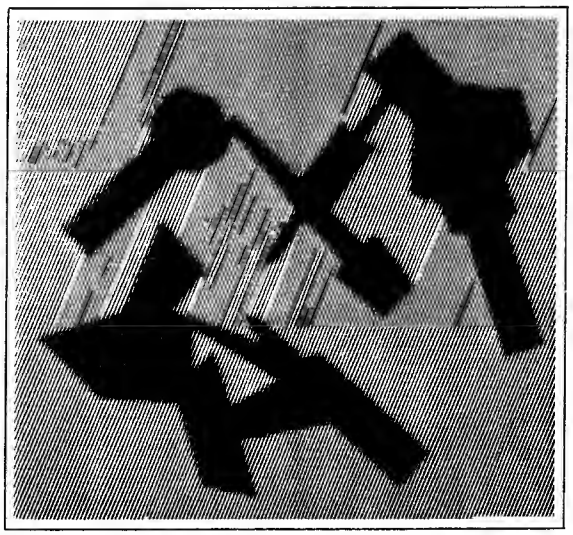

Figure 9.13: Another image from test set two.

of the number of total correct groups, and the number of correct groups of pairs of convex sections we would expect to get if we just combined convex sections at random. So we will carefully examine GROPER's performance on a typical image.

The image in figure 9.13, with edges in figure 9.14 provides a good example. First of all, GROPER succeeds in recognizing one of the objects in the image using only a single convex section of edges. After removing the edges that object accounts for from the image, GROPER divides the remaining edges into 33 convex sections. Ten of the thirty pairs of sections GROPER finds most likely to come from a single object really do. Figure 9.15 shows some of these ten groups, and figure 9.16 shows a few incorrect groups in the top thirty. At this point, the image actually does have eleven different pairs of convex sections that really come from a single object. Since the image has a total of 512 different pairs of convex sections, if we chose thirty pairs at random, the expected number of correct pairs we would find would be .65. So we can see that the distance and orientation constraints allow GROPER to combine pairs of convex sections very well, and much better than at a random rate.

The performance of the grouping system drops somewhat in tests three and five. We might expect that, because these images contain more objects than do the others. And test five has many more edges that do not correspond to object boundaries. These factors produce many more convex sections in images, and increase the chances that some convex sections will appear to go together well, even though they do not come from a single

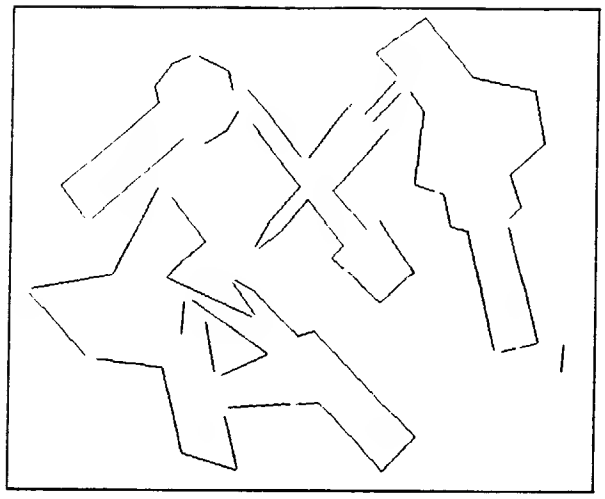

Figure 9.14: The edges it produces.

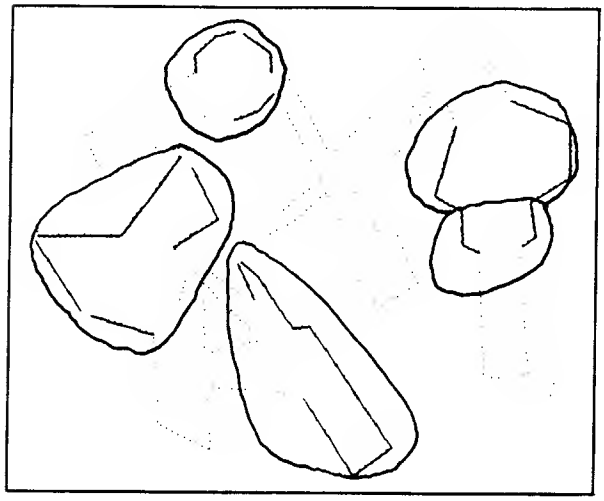

Figure 9.15: Some of the correct groups GROPER finds in the image, out of the thirty it picks first. Each pair of convex sections is circled.

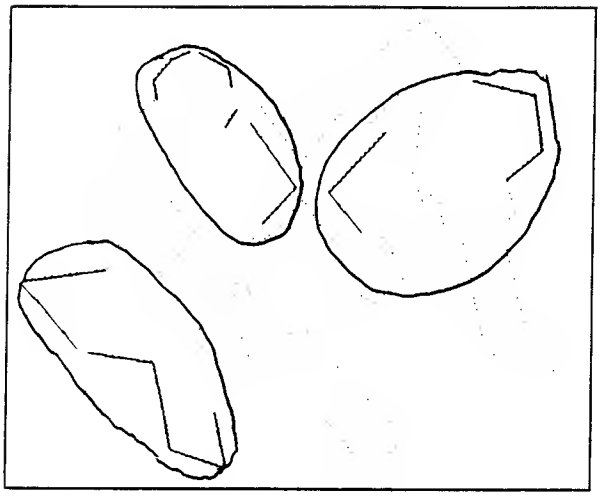

Figure 9.16: Some of the incorrect groups GROPER finds. The edges in each group are circled.

object. These factors would also make a random search for objects take much longer. And GROPER's performance on these images does still seem good. Figures 9.17 and 9.18 show some of the first groups GROPER picks from the image shown in figure 9.6.

GROPER's grouping system begins to have grave problems when confronted with an image like the one in figure 9.8. This image produces a great many edges that do not come from occlusions. And objects fade into their background much more, making it harder to form good convex sections of edges. In fact, the edges produced by this image form a fairly chaotic picture, even for a person. This tells us that we can not take straight line approximations to the edges in a realistic image, and expect to perform the kind of perceptual grouping on which GROPER relies. For this type of image, we must either develop much better techniques for finding the occluding edges in an image, or take quite a different approach to grouping, an approach that does not rely solely on edge information.

9.3 Results of Indexing

We want to test GROPER's indexing system to determine its space requirements, its accuracy, and its efficiency. In terms of accuracy, we have designed the indexing system so that it will always tell us all the collections of model edges that could match a set of image edges, except for bugs, of course. But we also want to know how often it tells us about collections of model edges that could not really have come from those image edges. These types of errors can have two possible effects. First of all, GROPER decides whether to try to verify matches based on the number of sets of model edges that match a particular set

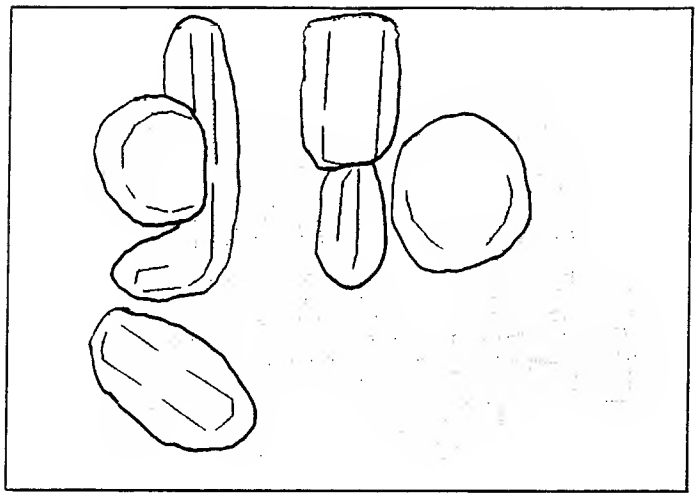

Figure 9.17: The correct groups GROPER finds in the image in figure 9.6, out of the thirty it picks first. Each is circled.

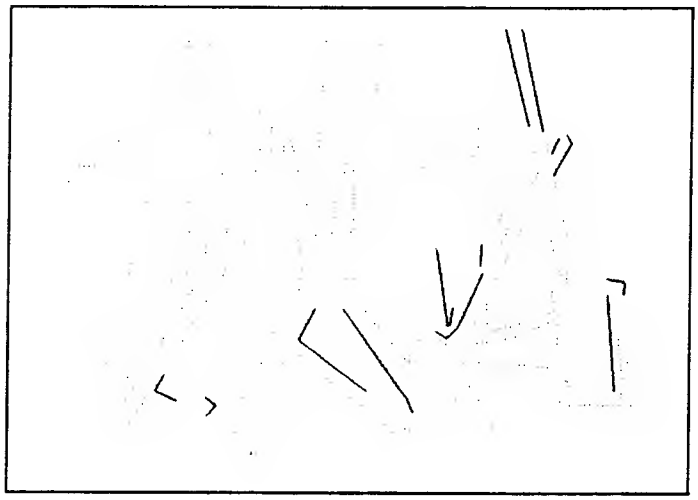

Figure 9.18: Some of the incorrect groups GROPER finds. Each group is circled.

Object Number	1	2	3	4	5	6	7	8
Number of Edges	6	10	8	8	9	9	13	9
Number of Table Entries	7,317	14,719	10,186	13,361	15,735	6,945	20,729	12,578
Object Number	9	10	11	12	13	14	15	16
Number of Edges	11	20	18	31	16	11	8	8
Number of Table Entries	30,209	66,586	20,741	161,388	17,465	17,379	9,049	11,276

Figure 9.19: The number of table entries for each of the sixteen models shown in figure 9.2.

of image edges. If we get too many false positive matches per indexing step, GROPER will not attempt to perform verification in situations where it should. This will effect the accuracy with which GROPER recognizes objects. Secondly, GROPER indexes with large collections of image edges by performing a series of table lookups for the pairs of image edges, and combining the results with intersections. If most of the matches found by a table lookup could not come from the pairs of image edges we are looking up, then most of the work required to combine these results is wasted. This will effect the speed with which GROPER recognizes objects. So we have collected some statistics that allow us to determine the amount of space required for indexing, as well as its impact on GROPER's speed and accuracy.

GROPER uses a lot of space. We know from its construction that GROPER's indexing table will require order Σn^2 table entries, where n ranges over the number of edges in each object in the library. This number becomes large. Moreover, GROPER makes a large number of table entries for each pair of edges. Figure 9.19 provides information about storage requirements. For the library of the sixteen objects shown in figure 9.2, grouping had to make over 400,000 table entries. This resulted in overall storage requirements significantly in excess of the available core memory of a Symbolics 3600 lisp machine. So recognition using this library became slow, as the random accessing of this indexing table caused constant paging.

These high space requirements result partly from the large number of entries required for each pair of edges. For example, the first object in figure 9.2 has six edges, and fifteen different pairs of edges. Yet it requires over 7,000 table entries, an average of almost 500

Number of Randomly Chosen Edges per Trial	2	3	4	5
Number of Trials	200	2,100	6,100	32,785
Average Number of Verifiable Matches per Trial	5.7	.40	.02	.0007
Average Number of Unverifiable Matches per Trial	25.06	3.05	.28	.03

Figure 9.20: Indexing was performed with random collections of edges. This table shows the number of correct positive responses, as well as the number of false positives.

entries per edge pair. As implemented, GROPER can handle a small number of objects that do not contain too many edges, like the libraries in figures 9.1 and 9.3. With some work, GROPER could probably handle somewhat more complex libraries. But the indexing system GROPER uses is not well suited to handling large libraries of objects.

Figure 9.20 presents the results of a second set of tests of GROPER's indexing system. To test GROPER, we first created random collections of edges. We did this by giving each edge an end point in a randomly chosen spot in a square 150 pixels wide. We then selected a random length, and random angle for each edge. We performed indexing with each group of edges, to see how many matches GROPER would find. For this test we used a lookup table containing information about the sixteen objects shown in figure 9.2. Then for each match, we performed a verification step, described in chapter 7, to see if we could really find a single rotation and translation to align the matched model edges with the group of image edges. This tells us how often GROPER finds unverifiable matches per indexing step, and the ratio of verifiable to unverifiable matches found.

The results show that GROPER's indexing errors should not effect its accuracy in most cases. Usually, by combining two convex sections of edges GROPER obtains groups of four or more edges. An indexing step using four edges will, on the average, produce about .3 sets of matching model edges, as figure 9.20 shows. With five edges we get an average of less than .03 matches. This indicates that once we have a group of four or more image edges that come from the same object, we do not need to expect too many spurious matches to interfere with GROPER's attempt to identify the object. However, we do find that with a group of three image edges we get an average of about 3.05 matches. A lookup that produces more than four matches will cause GROPER to decide not to attempt verification. Furthermore, most of these matches do not actually pass a verification test. So a better indexing scheme, which only told us about verifiable matches, might well produce better

recognition results in this case.

The high rate of false positive matches found by indexing also has an effect on GROPER's speed. Recall that GROPER finds matches for large collections of edges by performing a separate indexing step for every pair of edges, and then combining the results. Three quarters of the matches produced by indexing with a pair of edges will not pass a verification test. So GROPER wastes much of the time it spends combining these results. This has not caused too much of a waste of time overall only because the effectiveness of grouping prevents GROPER from having to perform too many indexing steps.

GROPER's indexing system does prove adequate for its needs. It works quickly enough so that GROPER usually spends less than half its time indexing. And it provides results that do not for the most part interfere with GROPER's accuracy. However, a more accurate indexing system would probably work much faster. It would also require less storage space. For these results indicate that the majority of table entries for a pair of edges do not really correspond to possible values of the parameters that describe those edges. The simplifications made to allow us to easily build the lookup table have resulted in many inaccurate entries.

9.4 Results of Recognition

We tested GROPER's overall performance by comparing it to that of another recognition system called SEARCHER. SEARCHER works just like GROPER, using the same indexing and verification modules. It also uses the same criteria to decide whether or not a match indicates the presence of an object. However, SEARCHER does not use grouping. GROPER uses grouping to tell it which collections of edges to use to try to locate objects. Instead, SEARCHER performs a simple backtracking search through the space of all possible sets of edges. It does not have to explore this entire space. Like GROPER, when SEARCHER finds that some edges could not all come from any known object, SEARCHER does not consider any other collections of edges that include these. And, like GROPER, when SEARCHER recognizes an object it removes from consideration any edges that this object can explain. Since SEARCHER also has no way to tell the object side from the background side of an edge, it must consider both possibilities in its search, just as GROPER must impose figure/ground judgments on its groups. By replacing GROPER's guided search for objects with an undirected search, SEARCHER shows us exactly how much GROPER's grouping component adds to its performance.

We tested GROPER and SEARCHER on the four sets of images of two dimensional objects. Figure 9.21 summarizes the results of these tests. These results show GROPER always performing much better than SEARCHER, requiring relatively few indexing steps.

	Test 1	Test 2	Test 3	Test 4
Number of Known Objects in Each Image	9	8	16	6
Average Percentage of Objects Correctly Located by GROPER	84%	65%	60%	97%
Average Number of False Positive Identifications by GROPER	.8	1	5.7	0
Average Number of Edge Groups Considered by GROPER	98.8	75.2	452.3	18.6
Average Percentage of Objects Correctly Located by SEARCHER	58%	48%	31%	60%
Average Number of False Positive Identifications by SEARCHER	14	8.2	*16.5	5.2
Average Number of Edge Groups Considered by SEARCHER	25,984.6	5,805.6	*30,210	2981.6

Figure 9.21: The results of recognition experiments. *These figures do not include the results for one image because it took SEARCHER so long that it invariably caused the garbage collector on the Lisp Machine to crash.

However the results show a significant variation in GROPER's performance on the different sets of tests. We will first compare the performance of the two systems, and then discuss possible reasons why GROPER performs less well on certain types of images.

These experiments demonstrate the effectiveness of GROPER's grouping system both in reducing the number of indexing steps needed to find objects and also in improving the accuracy of the system. GROPER always requires far fewer indexing steps than SEARCHER does when the two systems attempt to find all the objects they can in an image. For example, with the second set of test images, SEARCHER performed approximately 75 times as many indexing steps, and with the fourth set of images it performed about 160 times as many. This greater efficiency clearly indicates the success of GROPER's grouping constraints in ordering the search through collections of edges.

We did not compare the total running time of the two systems, because we made no effort to optimize SEARCHER, and so it ran extremely slowly. And both systems took quite a long time to run in tests two and three, due to the constant paging the large library required. On the first set of tests, GROPER took an average of about three minutes and twenty seconds to locate all the objects it could, or an average of about twenty-two seconds for each known object in the scene. On the fourth set of images, GROPER took an average of one minute and thirteen seconds, or twelve seconds per object.

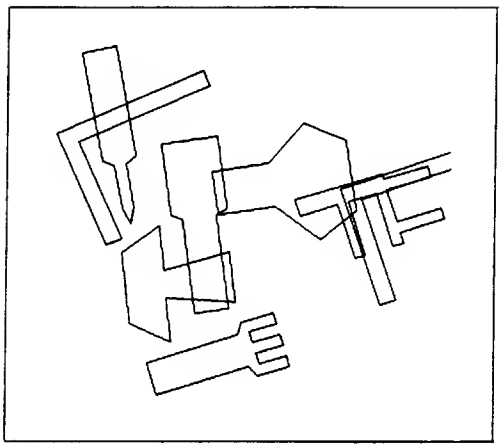

Figure 9.22: The objects GROPER found in the image in figure 9.4.

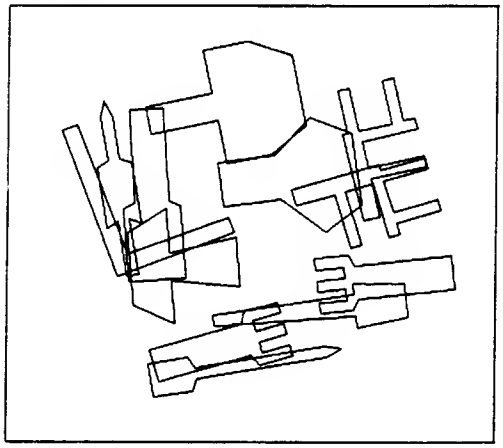

Figure 9.23: The objects SEARCHER found.

GROPER also consistently performed more accurately than searcher. It averaged 2.4, 1.4, 4.7 and 2.2 more objects correctly found on the four tests. And GROPER had few false positive identifications, except on the third set of tests, while SEARCHER always made quite large numbers of mistakes. Figures 9.22 and 9.23 show the objects that GROPER and SEARCHER found in the image in figure 9.4. However, we must approach these comparisons of accuracy a little skeptically, because we made a number of decisions regarding accuracy to optimize GROPER's effectiveness, not SEARCHER's.

We could have altered SEARCHER in two possible ways to improve its effectiveness. First of all, we could have increased the amount of computation SEARCHER needed, and improved its accuracy. Frequently SEARCHER would accept an identified object based on relatively little evidence, and then remove from future consideration edges that could

have helped provide much stronger evidence for a different object. So, instead of making use of the first verifiable match it found, SEARCHER could have looked in the image for all the objects it could find, and then accepted the best ones. This would have required more computation by removing one of the ways both it and GROPER prune their searches. Secondly, we use loose criteria for determining when to accept a match as valid. The two systems allowed an error of seven pixels in the location of any part of an object, and an error of $\frac{\pi}{10}$ in the angle between two edges. Furthermore, both systems accepted a match that accounted for as little as twenty-five percent of an object's perimeter. We used these criteria because with tighter criteria GROPER would occasionally miss an object, and even with these standards GROPER did not make many false positive identifications. But we could probably have improved SEARCHER's performance by tightening its standards for accepting a match. So we could certainly have made SEARCHER a more accurate recognition system by changing some parameters that we picked to optimize GROPER. But the fact that GROPER could get away with accepting the first valid matches it found, using loose standards, while SEARCHER could not, does indicate that grouping has helped make for a more accurate recognition system.

The reader should not, however, assume that SEARCHER typifies the performance of existing recognition systems. Certainly, many systems would perform more quickly and accurately than SEARCHER. But one way many systems do this is by using a certain amount of grouping. So the comparison with SEARCHER only reveals how much grouping helps GROPER, it does not tell us how GROPER would compare with existing recognition systems.

The tests run on GROPER show quite a variation in its accuracy. On the first set of tests, GROPER performed with good accuracy. But then on the second and third sets of images, GROPER performed much less well. These two sets of images used a different collection of objects. On closer examination of the results, we found that GROPER consistently failed to find certain objects in these images, while locating others successfully. For example, GROPER never found object six, and found object sixteen only one time in seven images. Figure 9.2 shows these objects. In general, GROPER often failed to locate objects that contained few parts, each of which had some short edges. The algorithm GROPER uses to make straight line approximations to edges usually eliminates these small edges, and they may also fail to appear due to the inherent difficulty of accurately locating edges at the corners of objects. Figure 9.11 and figure 9.12 provide an example of this problem. Without these short edges, GROPER has a hard time forming good convex sections to use in grouping. This would account for much of the difficulty GROPER has in detecting certain objects.

To check this hypothesis, we made a fourth set of tests using only six objects, which

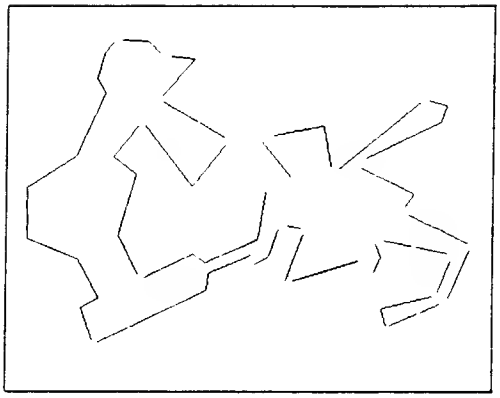

Figure 9.24: Straight line approximations to the edges produced by an image from test four.

GROPER had proven adept at recognizing. Figure 9.3 shows these six objects, and figure 9.24 shows the edges derived from one of the images in the test. The results proved encouraging, showing twenty-nine of thirty objects correctly identified, with no false positive identifications. In retrospect, this difficulty may account for many of the mistakes made in the first set of images as well. If the hypothesis is correct, we could overcome these problems by using low-level processing better suited to GROPER. GROPER does best when early processing gives it connected edges it can easily group together. We might try a straight line approximation algorithm that does not eliminate short edges that connect other edges. Or we might use the curved sections of edges themselves for grouping, avoiding the simplifications of straight line approximations.

The difficulties GROPER had in recognizing objects in tests two and three, as well as its difficulty in grouping on test six, do show certain inherent problems in GROPER's approach to grouping. This approach requires first finding meaningful sections of occluding edges in an image. These may be hard to come by in real scenes or in scenes with heavy occlusion. In those cases we would still expect the distance and orientation constraints to apply when appropriate. But other constraints will have to supplement them, helping us analyze sections of image where we can not find extended occluding edges.

9.5 Conclusions

This thesis has had three main goals. First, and foremost, we wanted to show the importance of using grouping to deal with difficult recognition tasks. Secondly, we wanted to show that we can effectively approach the grouping problem by looking at the image formation process, to see what it tells us about which parts of an image have the greatest

likelihood of coming from the same object. Finally, we wanted to show that we have successfully understood the influence of some important factors in grouping. The effectiveness of GROPER's grouping system helps to support each of these contentions.

To argue for the overall importance of grouping, we have maintained that when tackling complex recognition tasks, the greatest difficulties we encounter will be growing computational requirements and growing problems of accuracy. Handling large libraries of objects increases the number of possible matches between image features and object features. Furthermore, flexible objects and three dimensional objects will make it harder to eliminate incorrect hypotheses. We will need to match more of the image to the object before we can determine whether that match will work out, forcing us to consider more possibilities. And large libraries of flexible objects will create a much greater probability that some known object will have some way of matching any arbitrary collection of image edges, causing false positive identifications. GROPER has shown that grouping, especially when combined with indexing, can greatly reduce the combinatorics of recognition. Instead of searching through a huge number of possible matches, we can select a few groups of image edges, and quickly determine which objects might have produced them. We can use grouping to improve the accuracy of a recognition system because it provides us with an additional source of information to use when evaluating matches between the image and the object models. Instead of just looking at whether we can align an object model with some image edges, we can also look at whether those image edges seem to all come from the same object. The mistakes SEARCHER makes show that even with relatively simple recognition tasks, we can not reliably base our decision about whether to accept a match purely on the basis of how well a model aligns with image edges. So we have argued for the use of grouping by pointing out potential difficulties in extending current recognition systems to handle harder problems, and by demonstrating that grouping can help alleviate these problems.

We also wanted to show that one need not resort to completely ad-hoc solutions to the problem of grouping. Gibson (Gibson[8], for example) has persuasively argued for the importance in vision research of understanding constraints provided by the physical world and the image formation process. Marr and his followers (Marr[20], for example) have effectively applied this viewpoint in computer vision research. In this thesis we have attempted to capitalize on this point of view in analyzing the problem of grouping. We have attempted to understand the factors in the image formation process that determine the influence of distance and orientation on the likelihood that sections of an image originate with the same object. We have presented only a rudimentary analysis of this problem. Yet this analysis has still provided insights that have helped to build an accurate grouping system. And the success of GROPER's grouping system provides some evidence that this analysis is at least on the right track.

Appendix A

Making Random Objects

We have performed empirical tests to find a probability distribution for the lengths of occlusions that random objects produce. Chapter 3 reports the results of these tests. Performing these experiments required the construction of random objects. This appendix will discuss some problems involved in building random objects, and how we built them for these experiments.

We built random objects out of randomly constructed convex parts. To make a random convex shape, we selected the size of the shape, and its number of edges, according to a probability distribution. We then used a simple algorithm to construct a connected, closed shape that would meet these two criteria.

We wanted particularly to make sure that we constructed shapes with a random size, because this seemed like the factor most likely to influence the size of the occlusions produced by the object. As chapter 3 mentions, we expect large objects to produce longer occlusions than short ones. So we selected the area of the convex shape from a uniform random distribution from 0 to an arbitrary constant. Notice that we would have gotten different results by choosing the perimeter of the object from a uniform distribution, or by randomly choosing the distance from the viewer to the object. These choices would have produced smaller objects, because the area of an object increases with the square of its perimeter. So, since we had no convincing reason for choosing one over the other, we picked the size distribution that would tend to produce the worst results for our hypothesis that short occlusions occur more often then do long ones.

To determine the number of sides of a convex shape we picked a number between three and seven, with each number equally probable.

Then, to construct a random convex polygon, we started with a randomly chosen triangle, and added edges, one at a time, until we had enough. To make a random triangle, we chose a base with a random length. We then picked a number between 0 and π for the

angle between the base and one edge, and we picked a number between 0 and π minus the first number for the angle between the base and the second edge. The edge length and the two angles define the triangle. We then recursively added one new edge at a time to this polygon. To add an edge, we first picked one of the polygon's edges at random, and removed it. We connected the polygon again by adding two randomly chosen edges, subject to the constraint that they must maintain the convexity of the polygon. Having created a random convex polygon with the appropriate number of sides, we then scaled the polygon so that it would have the appropriate area.

We conducted experiments using objects with many convex parts. To form an object with a specific number of parts, we created each convex part separately, and then randomly joined them together. To connect a convex part to an object, we randomly located each of them in an image sized space. If they intersected, we joined them at the points of intersection, and removed all overlapping material. If they did not intersect, we tried randomly locating them again, until they did intersect.

This method of forming random objects has the disadvantage that it makes it difficult to characterize the distribution of all the variables that describe a random object. For example, we can not easily determine the distribution of the lengths of edges in such a random object. We can not even easily tell how many edges a random object has, since joining two convex parts together may eliminate some of the edges, or it may not. However, even if we could easily characterize these distributions, we would have no principled basis for choosing one distribution over another. None of these distributions would describe the real world, which does not contain random objects at all. So instead, we pick an arbitrary "random" world, which we can easily test. We do not expect that different kinds of random objects would produce dramatically different results. In any event, we do not intend the experiments that use these objects to prove anything about general random worlds. Rather, we perform them to gain confidence in our intuitions about how occlusions occur in the real world, and why short occlusions will occur more often than long ones.

Appendix B

The Likelihood of a $type_2$ Orientation

This appendix derives the probability of a $type_2$ orientation occurring between randomly oriented groups, in a simplified situation. We suppose that each group is infinitesimally small. We then calculate the probability that the projections of the two groups intersect. Chapter 3 provides the background needed to understand this appendix, and makes use of the results.

We begin with some definitions. Call the two groups A and B. Since they are infinitesimal, A and B occur on points in the plane, which we will also call A and B. A and B both have projections. If finite, these projections will actually be infinitesimal also, and only cover either point A or point B. If infinite, this projection will have an angle between 0 and π. We will call the angles of A's projection α, and of B's projection β. Two ray's will bound A's projection. We will call them a_1 and a_2. b_1 and b_2 will refer to the rays that bound B's projection. Without loss of generality, we can assume that A lies at the origin, and B lies on the x axis. We will call the angle between a_1 and the x axis θ, and the angle between b_1 and the x axis ψ. Figure B.1 illustrates these variables.

If either group has a finite projection, we can easily produce an answer. If group A has a finite projection, then the two groups' projections intersect only when point A lies inside B's projection. Since B's projection has a uniform, random orientation, it will have a probability of $\frac{\beta}{2\pi}$ of covering point A. Similarly, if B has a finite projection there will be a probability of $\frac{\alpha}{2\pi}$ of a $type_2$ orientation occurring. And of course, if both groups have finite projections, they can not intersect.

We now assume that both groups have infinite projections. We will determine the probability that these projections intersect by dividing the problem into eight different parts. First of all, we will consider separately the cases when $\alpha < \frac{\pi}{2}$ and when $\alpha > \frac{\pi}{2}$.

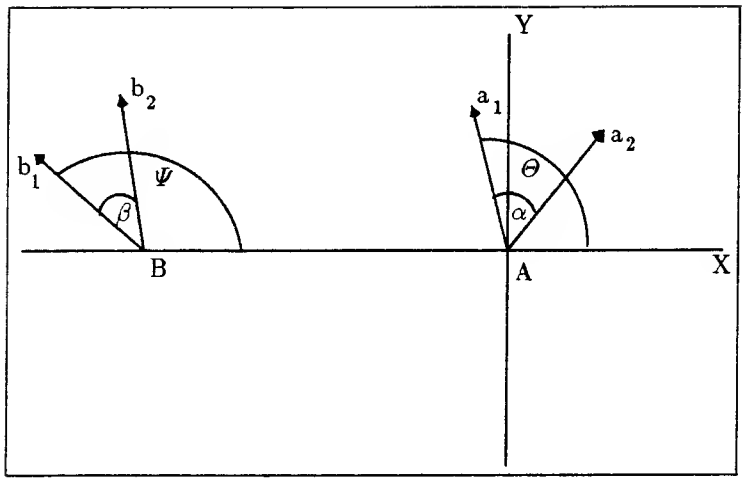

Figure B.1: Two tiny groups with infinite projections.

Then for each of those cases, we will divide the possible orientations of A's projection into four groups, depending on the quadrant in which the ray a_1 falls.

First, suppose $\alpha < \frac{\pi}{2}$.

Suppose that a_1 falls in the quadrant where x is negative and y is positive, that is, $\frac{\pi}{2} < \theta < \pi$. This tells us first of all that all of A's projection falls in the half-plane where y is positive. So B can not fall in A's projection. In fact, the projections intersect only when either A falls in B's projection, or when b_2 intersects a_1. This happens if $\psi - \beta < \theta$, i.e. if $\psi < \theta + \beta$. This occurs whenever ψ has a value between 0 and $\frac{\pi}{2} + \beta$. Furthermore, since θ has a uniform distribution between $\frac{\pi}{2}$ and π, the projections will also have a fifty percent chance of intersecting if ψ has a value between $\frac{\pi}{2} + \beta$ and $\pi + \beta$. So the total probability of intersecting projections is:

$$\frac{\frac{\pi}{2} + \beta + \frac{\pi}{4}}{2\pi}$$

Suppose that a_1 falls in the quadrant where x and y are both positive, i.e. that $\theta < \frac{\pi}{2}$. We must consider two possibilities. A's projection may all fall in one quadrant. This happens when $\alpha < \theta$. Or, A's projection may fall in two quadrants. In the first case, the projections will intersect whenever $0 < \psi$ and $\psi - \beta < \theta$. As θ has equal chances of having any value from α to $\frac{\pi}{2}$, the chances that ψ achieves an appropriate value are: $(\beta + \frac{\alpha + \frac{\pi}{2}}{2}) * \frac{1}{2\pi}$. Furthermore, the situation where $\alpha < \theta$, occurs with probability $1 - \frac{\alpha}{\frac{\pi}{2}}$. So, the total probability of α being less than θ, and the projections intersecting, is:

$$(\beta + \frac{2\alpha + \pi}{4}) * (1 - \frac{2\alpha}{\pi}) * \frac{1}{2\pi}$$

In the second case, when $\theta < \alpha$, the projections intersect only when they intersect over an infinite space, that is, if b_1 or b_2 lie in between a_1 and a_2. This has a $\frac{\alpha+\beta}{2\pi}$ probability of occurring. So, overall, the probability that θ will be less than α, and that the two projections will then intersect is:

$$\frac{(\alpha+\beta)}{2\pi} * \frac{2\alpha}{\pi} = \frac{2\alpha^2 + 2\alpha\beta}{2\pi^2}$$

Combining these two results, we find that, if $\theta < \frac{\pi}{2}$, the probability of projections intersecting is:

$$((\beta + \frac{2\alpha+\pi}{4}) - (\frac{8\alpha\beta + 4\alpha^2 + 2\alpha\pi}{4\pi}) + (\frac{2\alpha^2+2\alpha\beta}{\pi})) * \frac{1}{2\pi}$$

$$= (\beta + \frac{\alpha}{2} + \frac{\pi}{4} - \frac{2\alpha\beta}{\pi} - \frac{\alpha^2}{\pi} - \frac{\alpha}{2} + \frac{2\alpha^2}{\pi} + \frac{2\alpha\beta}{\pi}) * \frac{1}{2\pi}$$

$$= (\beta + \frac{\pi}{4} + \frac{\alpha^2}{\pi})\frac{1}{2\pi}$$

Next, suppose a_1 falls in the quadrant where x is positive and y negative. Once again, B can not fall in A's projection. In this case, a_2 makes an angle of $\alpha + (2\pi - \theta)$ with the x axis. An intersection occurs if B's projection extends anywhere within this range. This happens if $\theta - \alpha < \psi$. If $\psi < \beta$, then A falls inside B's projection. So, the total chances of projections intersecting are:

$$(\frac{\pi}{4} + \alpha + \beta)\frac{1}{2\pi}$$

Finally, suppose a_1 falls in the quadrant where x and y are both negative. First of all, B may fall in A's projection, with probability $\alpha\frac{2}{\pi}$. Secondly, if this does not happen, an intersection occurs if $\psi > \theta - \alpha$, or if $\psi < \beta$. Given that B does not fall in A's projection, θ will randomly range from $\frac{3\pi}{2}$ to $\frac{3\pi}{2} - \alpha$. So an intersection occurs when ψ falls in a range that varies between π to 2π and $\frac{3\pi}{2} - \alpha$ to 2π, or ψ may fall in the range from 0 to β. Overall, the chances of this happening are:

$$\frac{(\pi+\beta) + (\frac{\pi}{2} + \beta + \alpha)}{2} * \frac{1}{2\pi} * \frac{\frac{\pi}{2} - \alpha}{\frac{\pi}{2}}$$

When we add in the chances of A's projection including B, we find that the total probability of the projections intersecting is:

$$(\frac{\frac{3\pi^2}{4} - \frac{3\alpha\pi}{2} + \pi\beta - 2\beta\alpha + \frac{\pi\alpha}{2} - \alpha^2}{\pi} + 4\alpha)\frac{1}{2\pi}$$

$$= (\frac{3\pi}{4} + 3\alpha + \beta - \frac{2\beta\alpha}{\pi} - \frac{\alpha^2}{\pi})\frac{1}{2\pi}$$

We now know, if $\alpha < \frac{\pi}{2}$, the probability of the projections of A and B intersecting, depending on the quadrant in which a_1 lies. Since a_1 is equally likely to fall in any of these quadrants, we can just average these probabilities, to find the total likelihood that the two projections intersect. Doing this, we get:

$$\frac{\frac{\pi}{2} + \beta + \alpha - \frac{\alpha\beta}{\pi}}{2\pi}$$

Similar, tedious reasoning produces the same result when $\alpha > \frac{\pi}{2}$.

Appendix C

Lower *types* and Whether Groups Come from the Same Object

This appendix shows that, all other factors being equal, if two groups have a lower *type* of orientation they will have a greater chance of coming from the same object.

We can prove this mathematically. First of all, we will prove that:

$$\frac{P(type_1|O_1 = O_2, rd)}{P(type_1|O_1 \neq O_2, rd)} > \frac{P(type_2|O_1 = O_2, rd)}{P(type_2|O_1 \neq O_2, rd)}$$

(In this appendix we abbreviate *restofdata* with "*rd*", to make some long formulas more readable.) The ratio on the left indicates the chances that the two groups come from different objects when they have a $type_1$ relationship. The ratio on the right indicates the same thing for a $type_2$ relationship. So the inequality asserts that, given the same data, there is a greater chance the groups come from different objects when they have a $type_2$ relationship.

$$\frac{P(type_1|O_1 = O_2, rd)}{P(type_1|O_1 \neq O_2, rd)} =$$

$$\frac{P(type_1|same, O_1 = O_2, rd) * P(same|O_1 = O_2, rd)}{P(type_1|O_1 \neq O_2, rd)}$$

$$+ \frac{P(type_1|adj, O_1 = O_2, rd) * P(adj|O_1 = O_2, rd)}{P(type_1|O_1 \neq O_2, rd)}$$

$$+ \frac{P(type_1|notadj, O_1 = O_2, rd) * P(notadj|O_1 = O_2, rd)}{P(type_1|O_1 \neq O_2, rd)}$$

where "same" stands for the two groups coming from the same convex section of the same object, and "adj" and "notadj" denotes that they come from adjacent or non-adjacent sections. Chapter 3 pointed out that $P(type_1|same, O_1 = O_2, rd) = 1$, and described the

assumption that groups from different sections of the same object have random orientations, subject to the constraint that groups from adjacent sections can not produce a $type_3$ orientation. That assumptions tells us that:

$$P(type_1|adj, O_1 = O_2, rd) = \frac{P(type_1|O_1 \neq O_2, rd)}{P(type_1|O_1 \neq O_2, rd) + P(type_2|O_1 \neq O_2, rd)}$$

and that:

$$P(type_1|notadj, O_1 = O_2, rd) = P(type_1|O_1 \neq O_2, rd)$$

From this we find:

$$\frac{P(type_1|O_1 = O_2, rd)}{P(type_1|O_1 \neq O_2, rd)} =$$

$$\frac{P(same|O_1 = O_2, rd)}{P(type_1|O_1 \neq O_2, rd)}$$

$$+\frac{\frac{P(type_1|O_1 \neq O_2, rd)}{P(type_1|O_1 \neq O_2, rd)+P(type_2|O_1 \neq O_2, rd)} * P(adj|O_1 = O_2, rd)}{P(type_1|O_1 \neq O_2, rd)}$$

$$+\frac{P(type_1|O_1 \neq O_2, rd) * P(notadj|O_1 = O_2, rd)}{P(type_1|O_1 \neq O_2, rd)}$$

which equals:

$$\frac{P(same|O_1 = O_2, rd)}{P(type_1|O_1 \neq O_2)}$$

$$+\frac{P(adj|O_1 = O_2, rd)}{P(type_1|O_1 \neq O_2, rd) + P(type_2|O_1 \neq O_2, rd)}$$

$$+P(notadj|O_1 = O_2, rd)$$

Using similar reasoning we find that:

$$\frac{P(type_2|O_1 = O_2, rd)}{P(type_2|O_1 \neq O_2, rd)} =$$

$$\frac{\frac{P(type_2|O_1 \neq O_2, rd)}{P(type_1|O_1 \neq O_2, rd)+P(type_2|O_1 \neq O_2, rd)} * P(adj|O_1 = O_2, rd)}{P(type_2|O_1 \neq O_2, rd)}$$

$$+\frac{P(type_2|O_1 \neq O_2, rd) * P(notadj|O_1 = O_2, rd)}{P(type_2|O_1 \neq O_2, rd)}$$

which equals:

$$\frac{P(adj|O_1 = O_2, rd)}{P(type_1|O_1 \neq O_2, rd) + P(type_2|O_1 \neq O_2, rd)}$$

$$+P(notadj|O_1 = O_2, rd)$$

Clearly:

$$\frac{P(same|O_1 = O_2, rd)}{P(type_1|O_1 \neq O_2)}$$

$$+\frac{P(adj|O_1 = O_2, rd)}{P(type_1|O_1 \neq O_2, rd) + P(type_2|O_1 \neq O_2, rd) + P(notadj|O_1 = O_2, rd)}$$

$$> \frac{P(adj|O_1 = O_2, rd)}{P(type_1|O_1 \neq O_2, rd) + P(type_2|O_1 \neq O_2, rd) + P(notadj|O_1 = O_2, rd)}$$

and so the inequality holds. Similar reasoning tells us that:

$$\frac{P(type_2|O_1 = O_2, rd)}{P(type_2|O_1 \neq O_2, rd)} > \frac{P(type_3|O_1 = O_2, rd)}{P(type_3|O_1 \neq O_2, rd)}$$

This reasoning shows that our theory of grouping predicts that, all other factors being equal, groups with lower types of orientations have a greater chance of coming from the same object.

Appendix D

Arbitrary Constants Used in GROPER

GROPER contains a number of arbitrary constants. We chose these constants before performing the tests described in chapter 9. This appendix collects them, for convenience.

The maximum object diameter equals 300 pixels. GROPER uses this in determining the probability distribution of distances that separate two groups. Chapter 5 describes this in more detail.

The probability that parts of two different objects intersect equals .2. GROPER uses this in determining the probability that a certain distance separates two groups that come from different objects. As chapter 5 describes, GROPER finds this probability using a linear combination of two probability distributions. This constant determines how GROPER combines them.

When an indexing step produces 4 or fewer matches, GROPER performs verification on each match.

The probability that two groups from the same object come from the same convex section of that object, adjacent convex sections, or non-adjacent convex sections = $\frac{1}{3}$. GROPER uses this to determine the likelihood of different *type*'s of orientations occurring.

The maximum diameter of a convex part of an object equals 150 pixels. If two groups have projections that intersect, but the distance from one group to the intersection of their projections exceeds 150 pixels, then GROPER decides they cannot have a $type_2$ relationship. In general, GROPER uses this value to decide how int_1 and int_2 effect the likelihood of two groups coming from adjacent sections of the same object.

GROPER performs indexing with single convex sections of edges only if they contain at least four edges.

If indexing shows that a pair of convex sections match many different sets of object edges,

GROPER next explores the five best triples of groups obtained by adding an additional group.

GROPER allows a maximum error of 7 pixels in the sensed location of any edge in the image.

The maximum error expected in the sensed angle of an edge is $\sin \frac{\pi}{10}$.

The lookup table for indexing quantizes distances to a multiple of 20 pixels.

And it quantizes angles to a multiple of $\frac{\pi}{16}$.

GROPER accepts a match if it accounts for at least 25% of an object's perimeter.

Bibliography

[1] Biederman, I., 1987. "Matching Image Edges to Object Memory." *Proceedings First International Conference on Computer Vision*: 384-392.

[2] Binford, T., 1982. "Survey of Model-Based Image Analysis Systems." *The International Journal of Robotics Research*, 1(1):18-64.

[3] Besl, P. and Jain, R., 1985. "Three-Dimensional Object Recognition." *ACM Computing Surveys*, 17(1):75-145.

[4] Bolles, R. and Cain, R., 1982. "Recognizing and Locating Partially Visible Objects: The Local-Feature-Focus Method." *The International Journal of Robotics Research*, 1(3):57-82.

[5] Brooks, R., 1981. "Symbolic Reasoning Among 3-D Models and 2-D Images." *Artificial Intelligence*, 17:285-348.

[6] Carlotto, M., 1987. "Histogram Analysis Using a Scale-Space Approach." *IEEE Transactions on Patern Analysis and Machine Intelligence*, 9(1):121-129.

[7] Ettinger, G., 1988. "Large Hierarchical Object Recognition Using Libraries of Parameterized Model Sub-parts." *Proceedings, Conference on Computer Vision and Pattern Recognition '88*:32-41.

[8] Gibson, J., 1979. *The Ecological Approach to Visual Perception.* Boston: Houghton Mifflen.

[9] Grimson, W.E.L. and Lozano-Pérez, T., 1984. "Model-Based Recognition and Localization from Sparse Range or Tactile Data." *The International Journal of Robotics Research*, 3(3):3-35.

[10] Grimson, W.E.L. and Lozano-Pérez, T., 1987. "Localizing Overlapping Parts by Searching the Interpretation Tree." *IEEE Transactions on Patern Analysis and Machine Intelligence*, 9(4):469-482.

[11] Gross, A. and Rosenfeld, A., 1987. "Multiresolution Object Detection and Delineation." *Computer Vision, Graphics, and Image Processing*, 39:102-115.

[12] Haralick, R. and Shapiro, L., 1985. "Image Segmentation Techniques." *Computer Vision, Graphics, and Image Processing*, 29:100-132.

[13] Huttenlocher, D., 1988. *Three-Dimensional Recognition of Solid Objects from a Two-Dimensional Image.* PhD. Dissertation, MIT Dept. of Elect. Eng. and Computer Sci.

[14] Kalvin, A., Schonberg, E., Schwartz, J., and Sharir, M., 1986. "Two-Dimensional, Model-Based, Boundary Matching Using Footprints." *The International Journal of Robotics Research*, 5(4):38-55.

[15] Knoll, T. and Jain, R., 1986. "Recognizing Partially Visible Objects Using Feature Indexed Hypotheses." *IEEE Journal of Robotics and Automation*, 2(1):3-13.

[16] Kohler, W. 1959. *Gestalt Psychology.* New York: Mentor Books.

[17] Land, E. and McCann, J., 1971. "Lightness and Retinex Theory." *Journal of the Optical Society of America*, 61:1-11.

[18] Lowe, D., 1985. *Perceptual Organization and Visual Recognition.* The Netherlands: Kluwer Academic Publishers.

[19] Mahoney, J., 1987. *Image Chunking: Defining Spatial Building Blocks for Scene Analysis.* M.I.T. AI-TR 980.

[20] Marr, D., 1982. *Vision.* New York: W.H. Freeman and Company.

[21] Pavlidis, T., 1972. "Segmentation of Pictures and Maps through Functional Approximation." *Computer Vision, Graphics, and Image Processing*, 1:360-372.

[22] Schwartz, J. and Sharir, M., 1987. "Identification of Partially Obscured Objects in Two and Three Dimensions by Matching Noisy Characteristic Curves." *The International Journal of Robotics Research*, 6(2):29-44.

[23] Thompson, D. and Mundy, J. 1987. "Three-Dimensional Model Matching from an Unconstrained Viewpoint." *Proceedings of the IEEE Conference on Robotics and Automation*,208-220.

[24] Turney, J., Mudge, T., and Volz, R., 1985. "Recognizing Partially Occluded Parts." *IEEE Transactions on Patern Analysis and Machine Intelligence*, 7(4):410-421.

[25] Wallace, A., 1987. "Matching Segmented Scenes to Models Using Pairwise Relationships Between Features." *Image and Vision Computing*, 5(2):114-120.

[26] Wertheimer, M., 1938. "Laws of Organization in Perceptual Forms." *A Source Book of Gestalt Psychology*,71-88. Edited by Ellis, W. New York: Harcourt, Brace and Company.

[27] Witkin, A. and Tenenbaum, J., 1983. "What is Perceptual Organization for?" *Proceedings of the Eight International Joint Conference on Artificial Intelligence*:1023-1027.

[28] Yang, H. and Kak, A., 1986. "Determination of the Identity, Position and Orientation of the Topmost Object in a Pile: Some Further Experiments" *Proceedings of the IEEE Conference on Robotics and Automation*,293-298.

CS-TR Scanning Project
Document Control Form

Date : 9/14/95

Report # AI-TR-1023

Each of the following should be identified by a checkmark:
Originating Department:

- [x] Artificial Intellegence Laboratory (AI)
- [ ] Laboratory for Computer Science (LCS)

Document Type:

- [x] Technical Report (TR)
- [ ] Technical Memo (TM)
- [ ] Other:_____

Document Information

Number of pages: 162 (170-IMAGES)
Not to include DOD forms, printer instructions, etc... original pages only.

Originals are:

- [x] Single-sided or
- [ ] Double-sided

Intended to be printed as :

- [ ] Single-sided or
- [x] Double-sided

Print type:

- [ ] Typewriter
- [ ] Offset Press
- [x] Laser Print
- [ ] InkJet Printer
- [ ] Unknown
- [ ] Other:_____

Check each if included with document:

- [x] DOD Form (2)
- [ ] Funding Agent Form
- [x] Cover Page
- [x] Spine
- [ ] Printers Notes
- [ ] Photo negatives
- [ ] Other: _____

Page Data:

Blank Pages(by page number):_____

Photographs/Tonal Material (by page number): 12, 132-134, 136-137

Other (note description/page number):

Description : (162) Page Number:

IMAGE MAP: PAGES #'ED 1-162 (INCLUDING TITLE PAGE)
(163-170) SCANCONTROL, COVER, SPINE, DOD(2)
TRGT'S (3)

Scanning Agent Signoff:

Date Received: 9/14/95 Date Scanned: 9/18/95 Date Returned: 9/21/95

Scanning Agent Signature: Michael W. Cobb

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AI-TR 1023	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) The Use of Grouping in Visual Object Recognition		5. TYPE OF REPORT & PERIOD COVERED technical report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) David W. Jacobs		8. CONTRACT OR GRANT NUMBER(s) N00014-86-K-0124 N00014-86-K-0685
9. PERFORMING ORGANIZATION NAME AND ADDRESS Artificial Intelligence Laboratory 545 Technology Square Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE October 1988 13. NUMBER OF PAGES 162
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research Information Systems Arlington, VA 22217		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution is unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

grouping
object recognition
indexing
model-based vision
libraries
segmentation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report explores the use of grouping in object recognition by computational systems. Many recognition systems in the past have performed an undirected search through the space of different segmentations of an image in order to recognize objects. This approach leads to significant problems of computational complexity and accuracy. The process of grouping determines the sections of an image most likely to come from a single object. This

over

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Block 20 (cont.)

can tell a recognition system which segmentations of the image to consider first, improving both its speed and accuracy.

The report describes a particular recognition system called GROPER. GROPER performs grouping by using distance and relative orientation constraints that estimate the likelihood of different edges in an image coming from the same object. The thesis presents both a theoretical analysis of the grouping problem and a practical implementation of a grouping system. And it discusses the relevance of this theory of grouping to human psychology. GROPER also uses an indexing module to allow it to make use of knowledge of different objects, any of which might appear in an image. We test GROPER by comparing it to a similar recognition system that does not use grouping.

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**